VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Klasifikácia s využitím prevodu vektorových dát na siete

# Classification Using Network Construction Methods

2020 Branislav Bellovič

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání diplomové práce

| | |
|---|---|
| Student: | **Bc. Branislav Bellovič** |
| Studijní program: | N2647 Informační a komunikační technologie |
| Studijní obor: | 2612T025 Informatika a výpočetní technika |
| Téma: | Klasifikace s využitím převodu vektorových dat na sítě<br>Classification Using Network Construction Methods |
| Jazyk vypracování: | čeština |

Zásady pro vypracování:

Cílem práce je implementace vybraných metod převodu vektorových dat na sítě a jejich využití pro klasifikaci. Preferován je jazyk C#.

1. Rešerše obdobných řešení.
2. Implementace vybraných metod konstrukce sítí z vektorových dat.
3. Implementace klasifikačních metod s využitím sítí a porovnání jednotlivých metod.
4. Implementace webové aplikace využívající implementované metody.
5. Dokumentace s využitím standardů softwarového inženýrství.

Seznam doporučené odborné literatury:

[1] Silva, T. C., Zhao, L. (2016). Machine learning in complex networks (Vol. 2016, p. 1). Switzerland: Springer.

Dále podle pokynů vedoucího práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Mgr. Miloš Kudělka, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020


doc. Ing. Jan Platoš, Ph.D.
*vedoucí katedry*

prof. Ing. Pavel Brandštetter, CSc.
*děkan fakulty*

Vyhlasujem, že som diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne parametre a publikácie, z ktorých som čerpal.

V Ostrave 15. mája 2020

Bellovič

Na tomto mieste by som chcel poďakovať doc. Mgr. Milošovi Kudělkovi, Ph.D. za odborné vedenie a pomoc pri vypracovaní diplomovej práce.

**Abstrakt**

Diplomová práca je zámerná na klasifikáciu vektorových dát pomocou sieti. Práca poskytuje prehľad problematiky spojenej s klasifikáciou vektorových dát pomocou sieti a detailnejšie sa zameriava na vplyv vytvorených sieti na výsledok klasifikácie. Hlavný cieľ tejto práce je vytvorenie aplikácie ktorá umožni prevod vektorových dát na sieť, využite siete na klasifikáciu a následne vyhodnotenie výsledku klasifikácie pomocou interaktívneho modelu siete. V práci budú vykonané experimenty ktoré ukážu že pre klasifikáciu sú vhodné predovšetkým riedke siete. Ďalej bude predstavených niekoľko algoritmov určených pre prevod vektorových dát na sieť, siete ktoré generujú a následne porovnanie výsledkov. V závere budú vykonané experimenty ktoré ukážu, že počet použitých vrcholov nutný pre dobré výsledky klasifikácie môže byť vypočítaný na základe vlastností siete.

**Kľúčové slová**: vektorové dáta, siete, konštrukcia sieti, klasifikácia, vizualizácia sieti

**Abstract**

This Master's thesis is focused on the classification of vector data using networks. The work provides an overview of the topics associated with the classification of vector data using networks and is primarily focused on the impact of the created networks on the result of the classification. The main goal of this work is to create an application that will allow to convert vector data to network, it classification and then the subsequent evaluation of classification results using an interactive network model. The experiments performed in this work will show that sparse networks are especially suitable for classification. Next, several network construction algorithms will be presented together with the networks they generate and then, subsequently, they will be compared. Finally, experiments will be performed to show that the number of vertices used fo classification that are necessary for good results can be calculated based on the properties of the network.

**Keywords**: vector data, networks, network construction, classification, network visualization

# Obsah

# Zoznam obrázkov

# Zoznam tabuliek

# 1 Úvod

Strojove učenie zastáva dôležitú pozíciu v oblasti informatiky. Dôležitosť tejto pozície je podložená skutočnosťou, že posledné desaťročie sa stalo desaťročím globalizácie a dát. Internet sa stal prepojenejším a obsahovo bohatším ako kedykoľvek predtým. Prepojenosť prináša mnoho obsahu vo forme dát, ktoré okrem hlavnej informácie nesú aj skrytú, na prvý pohľad neviditeľnú, informáciu. Táto informácia sa dá spracovať pomocou strojového učenia a ďalej využiť.

Strojové učenie zvyčajne využíva dáta vo vektorovej podobe, na ktoré aplikuje algoritmus. V posledných rokov sa však dostávajú do popredia techniky, ktoré sú založené na sieťach. Jednou z týchto techník je aj klasifikácia, ktorá je hlavnou témou tejto práce. Klasifikácia pomocou siete zahŕňa spracovanie dát vo vektorovej forme, zostrojenie grafu z vektorových dát a následne využitie grafu na klasifikáciu.

Úvodná kapitola tejto práce obsahuje krátky popis strojového učenia a následné zaradenie samotnej klasifikácie v oblasti strojového učenia. V nasledujúcich kapitolách je uvedené teoretické zázemie práce.
Na začiatku je venovaná pozornosť spracovaniu dát a ich transformácii do formy vhodnej na ďalšie použite. Takto spracované dáta sa zvyčajne nazývajú **dataset**. Štruktúra datasetu a techniky použité na spracovanie datasetu, ktoré majú vplyv na výsledok klasifikácie, sú uvedene v kapitole 3.
Ďalšia kapitola sa venuje definícii vzťahu medzi pozorovaniami, čiže jednotlivými záznamami datasetu. Tento vzťah je väčšinou definovaný funkciou podobnosti alebo vzdialenosti medzi jednotlivými záznamami.
V nasledujúcej kapitole je definovaný prevod datasetu na **sieť**. Sieť môže byť reprezentovaná rôznymi spôsobmi. Medzi najpopulárnejšie reprezentácie patrí matica susednosti alebo graf, ktoré sú bližšie popísane v 4. Okrem samotného prevodu sú v tejto kapitole uvedené niektoré algoritmy, ktoré tento prevod realizujú a boli použité v práci.
Posledná kapitola, s teoretickým zázemím, sa venuje klasifikácii pomocou sieti. Podobne ako v predchádzajúcej kapitole, je najskôr definovaný princíp klasifikácie a následne sú uvedené algoritmy, ktoré boli použité v práci.

Predposledná kapitola sa venuje aplikácii a knižniciam, ktoré boli implementované v rámci práce. V tejto kapitole sú v krátkosti popísané knižnice, ktoré boli implementované, a ich použitie v aplikácii doplnené o ilustračne zábery.

Posledná kapitola je venovaná experimentom nad implementovanými algoritmami. Experimenty sú zamerané na porovnanie klasifikácie 2 algoritmov implementovaných v práci s dôrazom na skúmanie vplyvu hustoty sieti vytvorených pomocou algoritmov, ktoré boli taktiež implementované ako súčasť práce.

# 2 Strojové učenie

Strojové učenie (angl. *Machine learning*) je podoblasť väčšieho celku informatiky s názvom umelá inteligencia. Ako bolo naznačené v úvode, strojové učenie prináša nové informácie o dátach pomocou aplikácie špecifických algoritmov. Algoritmy strojového učenia sa opierajú o štatistické metódy a snažia sa vyvodiť záver o dátach na základe ich vlastnosti a nie na základe explicitného postupu.

Keďže sa algoritmy strojového učenia snažia zistiť nejakú vlastnosť dát, o priebehu algoritmov hovoríme ako o **učení**. Algoritmy sa na základe vykonávanej úlohy rozdeľujú do množstva kategórií. Základné rozdelenie algoritmov je na algoritmy, ktoré sa učia s učiteľom (angl. *Supervised learning*) a algoritmy, ktoré sa učia bez učiteľa (angl. *Unsupervised learning*). Na pomedzí týchto 2 skupín sú algoritmy, ktoré sa učia s čiastočnou pomocou učiteľa (angl. *Semi-supervised learning*).

## 2.1 Učenie bez učiteľa

Učenie prebieha bez predošlej informácie o dátach. Cieľom algoritmov v tejto skupine je odhadnúť distribučnú funkciu, ktorou boli tieto dáta vytvorené. Medzi najznámejšie úlohy v tejto kategórii patria:

- **zhlukovanie** - cieľom zhlukovania je rozdeliť dáta do zhlukov (skupín), ktoré sa podobajú a zároveň dobre oddeliť dáta, ktoré patria do odlišných zhlukov.
- **detekcia odľahlých pozorovaní** (angl. *outlier detection*) - cieľom je nájsť pozorovania (objekty), ktoré sú veľmi odlišne od ostatných dát. Detekcia odľahlých pozorovaní býva často spájaná so zhlukovanim. Jej hlavným cieľom je odstrániť pozorovania ktoré nezapadajú do žiadneho zhluku.
- **dolovanie vzorov** (angl. *association rule mining*) - úlohou je hľadanie vzorov (angl. *pattern*) v dátach. Pri dolovaní vzorov je na rozdiel od zhlukovania kladený väčší dôraz na hľadanie vzťahu medzi pozorovaniami. Dolovanie vzorov sa dá použiť na pozorovania, u ktorých môžeme povedať, že nejakú vlastnosť buď majú, alebo nemajú.
- **redukcia dimenzionality** - s touto úlohou sa spája pojem "kliatba dimenzionality"(angl. *curse of dimensionality*), ktorá hovorí, že s rastúcou dimenziou sa pozorovania v datasete začínajú čím ďalej podobať medzi sebou a je veľmi obtiažne nájsť podobnosť medzi jednotlivými pozorovaniami. Vysoká dimenzionalita môže mať za následok prílišnú väzbu na konkrétne dáta, známu tiež pod pojmom *modle overfitting*, čo znamená, že algoritmus naučený na konkrétnych dátach nie je použiteľný na iné dáta z rovnakej distribúcie.
  Algoritmy pre redukciu dimenzionality, ako napovedá názov, minimalizujú počet dimenzií s ohľadom na zachovanie vlastnosti dát.

Redukcia dimenzionality prebieha často pred aplikovaním samotných algoritmov strojového učenia.

## 2.2 Učenie s učiteľom

Učenie s učiteľom, na rozdiel od učenia bez učiteľa, využíva informáciu, ktorá je priradená pozorovaniu. Táto informácia je zvyčajne označená ako štítok (angl. *label*) alebo trieda. Pozorovania, ktoré majú štítok, sú označované ako tréningové dáta a cieľom učenia je vytvoriť klasifikátor, funkciu, ktorá na základe atribútov a štítkov dokáže priradiť štítky testovacím dátam, čiže dátam, ktoré tento štítok nemajú. Ak je štítok diskrétna hodnota, úloha sa nazýva klasifikácia, a ak je štítok spojitá hodnota, úloha sa nazýva regresia.

Presnosť klasifikátora sa väčšinou overuje troma spôsobmi.

1. **holdout** - dataset je ručne rozdelený na trénovaciu a testovaciu kolekciu.

2. **k-násobná kros-validácia** (angl. *k-fold cross validation*) - princípom tejto metódy je rozdelenie datasetu na $k$ približne rovnako veľkých podmnožín tak, aby každá podmnožina mala rovnomerné zastúpenie každej triedy. Následne sa prevedie $k$ iterácií, pričom v každej iterácii sa označí jedna podmnožina ako testovacia kolekcia a ostatné sa zlúčia do tréningovej kolekcie. Po dokončení iterácie sa vypočíta priemerná úspešnosť, ktorá určuje presnosť klasifikátora. V praxi sa ukázalo, že najvhodnejšie $k$ pri dostatočnom množstve dát je $k = 10$.

3. **leave-one-out** - táto metóda je špeciálnym prípadom k-násobnej kros-validácie kde $k = L - 1$, pričom $L$ reprezentuje počet označkovaných objektov.

## 2.3 Učenie s čiastočnou pomocou učiteľa

Ako bolo naznačené, učenie s čiastočnou pomocou učiteľa sa nachádza na pomedzí medzi predošlými dvoma a kombinuje ich silné stránky. Typická úloha spočíva v propagácii štítkov z malého množstva označkovaných (angl. *labeled*) pozorovaní na veľké množstvo neoznačkovaných (angl. *unlabeled*).

Do tejto kategórie sa radia niektoré metódy klasifikácie pomocou siete, ktoré sú hlavnou témou tejto práce. Pred definíciou samotnej klasifikácie je však nutné definovať pojmy a úlohy, ktoré tejto klasifikácii predchádzajú.

# 3 Spracovanie datasetu

Dataset je dátová štruktúra podobná databázovej tabuľke, kde každý riadok reprezentuje jeden záznam a každý stĺpec reprezentuje jeden atribút záznamu. V kontexte strojového učenia sú jednotlivé záznamy tvorené pozorovaniami.

Atribúty rozdeľujem na základe ich typu nasledovne:

1. kvalitatívne
   (a) nominálne - ich hodnoty popisujú zaradenie do určitej kategórie alebo stavu. Nominálne atribúty sa nedajú zoradiť, majú však definovanú rovnosť, čiže je možné určiť či sa dve hodnoty rovnajú alebo nie.
   (b) ordinárne - majú definované poradie, vďaka čomu je možne používať operátory ako väčší, menší.
   (c) binárne - môžu nadobúdať 2 stavy / hodnoty.
2. kvantitatívne
   (a) numerické - majú numerickú hodnotu reprezentovanú celým alebo reálnym číslom. Numerické atribúty môžeme ďalej rozdeliť na intervalové a pomerové (angl. *ratio-scaled*). Intervalové atribúty majú hodnotu, ktorá sa dá porovnávať, ale nemajú referenčnú hodnotu nazývanú aj nulová hodnota. Hodnoty môžu byť sčítavané a odčítavané na definovanom intervale, ale nemôžu byť násobené alebo delené. Pomerové atribúty majú definované všetky aritmetické operácie.
   (b) spojité - majú nekonečne veľa hodnôt.
   (c) diskrétne - majú konečnú alebo spočítateľné nekonečnú množinu hodnôt.

Prvým krokom pred aplikáciou algoritmov strojového učenia je úprava záznamov. Tento proces je známy pod názvom čistenie dát (angl. *data cleaning*).

Medzi základné úkony čistenia dát patrí:

1. odstránenie duplicitných záznamov a irelevantných dát.
2. odstránenie štrukturálnych chýb - medzi tieto chyby patria napríklad preklepy, nekonzistentné použite začiatočných písmen slov (napr. $A$ vs $a$), alebo nekonzistentné označenie chýbajúcich hodnôt.
3. odstránenie odľahlých pozorovaní - odľahlé pozorovania môžu spôsobiť problémy pri niektorých modeloch. Vo všeobecnosti je potrebné, odstraňovať odľahlé pozorovania so zvýšenou pozornosťou.
4. spracovanie chýbajúcich hodnôt - na základe povahy chýbajúcej hodnoty je možno chýbajúcu hodnotu odstrániť alebo nahradiť.

5. prevod kategoriálnych dát na numerické - väčšina algoritmov nedokáže pracovať s kategoriálnymi dátami, a preto je potrebný ich prevod na numerické. Základnou technikou je transformácia kategoriálneho atribútu $n$ nových, pričom $n$ je počet kategórii, a následne u každého záznamu nastaviť príznak atribútu zodpovedajúcej kategórii.

6. normalizácia - v prípade dát nachádzajúcich sa v rôznych intervalov je nutné, aby boli normalizované. Pod pojmom normalizácia atribútu rozumieme prevod hodnôt atribútu na určitý interval. Tento interval je zvyčajne $< 0, 1 >$ alebo $< -1, 1 >$. Medzi základné techniky normalizácie patria:

   (a) z-normalizácia - transformácia hodnôt tak, aby priemer bol 0 a smerodajná odchýlka 1.[1]

   (b) min-max normalizácia - transformácia hodnôt na interval $< 0, 1 >$.[1]

# 4 Prevod vektorových dát na sieť

Siete nesú, prirodzene, väčšie množstvo informácie ako vektorové dáta. Vďaka prítomnosti topológie a štruktúry siete sú vzťahy medzí jednotlivými objektami systematicky usporiadané. Usporiadanie siaha od lokálnych vlastnosti až po globálne. Ďalej je tiež nutné dodať, že prevod vektorových dát na sieť je bezstratový, keďže vektorové dáta sú reprezentované jednotnými záznamami, medzi ktorými neexistuje prirodzený vzťah. Prevod zo siete na vektorové dáta je na druhej strane veľmi obtiažny a väčšinou nereálny, pretože reprezentácia vo forme vektorových dát nedisponuje mechanizmami, ako uchovať typické vlastnosti sietí vyplývajúce z už spomínanej štruktúry a topológie.

Vektorové dáta neobsahujú štrukturálne a topologické vlastnosti, preto je nutné použiť spôsob prevodu, ktorý dokáže sieť vytvoriť tak, aby jej štruktúra popisovala povahu predvádzaných dát.

Prevod vektorových dát na sieť prebieha typicky v troch krokoch:

1. **určenie podobnosti/vzdialenosti** záznamov datasetu - zvyčajne reprezentované maticou podobnosti alebo maticou vzdialenosti, ktorá je vytvorená pomocou vhodnej funkcie podobnosti alebo funkcie vzdialenosti.

2. **vytvorenie vrcholov** - pre každý záznam datasetu je vytvorený zodpovedajúci vrchol

3. **vytvorenie hrán** - hrany sú vytvorené pomocou procesu sparsifikácie matice definovanej v kroku 1. Matica vytvorená v kroku 1 je väčšinou takmer úplná, čiže väčšina záznamov obsahuje nenulovú hodnotu. Sparsfikáciu môžeme definovať ako maximalizačný problém, ktorého cieľom je dosiahnuť čo najväčší počet nulových hodnôt tak, aby nebola narušená štruktúra dát obsiahnutá v matici. V prípade prevodu vektorových dát na sieť pod pojmom sparsifikácie rozumieme aplikovanie algoritmov prevodu vektorových dát na sieť.

Výsledkom prevodu je matica podobnosti, ktorá môže byť následne prevedená na inú reprezentáciu. Medzi základne spôsoby uchovania siete patrí:

1. **matica susedstva** - ide o maticu veľkosti $A = N \times N$, kde $N$ je počet vrcholov. Riadky aj stĺpce reprezentujú vrcholy siete a hodnoty reprezentujú hrany medzi vrcholmi. Matica susedstva môže byť vážená alebo nevážená. Ak je matica nevážená a bez násobných hrán, hrana medzi vrcholmi je reprezentovaná celým číslom 1 alebo príznakom *true*. Ak sieť obsahuje násobné hrany, počet hrán je reprezentovaný celým číslom. V prípade váženej matice je hrana reprezentovaná reálnym číslom, ktoré reprezentuje váhu.

2. **Graf** - Graf je definovaný ako usporiadaná dvojica $G(V, E)$, kde $V$ je neprázdna množina vrcholov a $E$ je dvoj-prvková podmnožina množiny V.

3. **Zoznam susednosti** - jednoduchá štruktúra, v ktorej každý riadok reprezentuje hranu medzi dvojicou vrcholov.

Použitie vhodnej funkcie na porovnávanie záznamov v kombinácii s vhodným algoritmom prevodu má kľúčovú úlohu pri tvorbe štruktúry.

## 4.1 Miery podobnosti záznamov

Podobnosť a odlišnosť zastávajú dôležitú úlohu v strojovom učení. Vo všeobecnosti slúži podobnosť a odlišnosť na porovnávanie dvoch objektov. V praxi sú tieto dva pojmy často zamieňané a aj keď slúžia na rovnaký účel, existujú medzi nimi isté rozdiely.

### 4.1.1 Podobnosť

Nech $X$ je neprázdna množina objektov s definovaným vzťahom rovnosti. Ak $s$ je funkcia podobnosti, potom je $s$ zhora ohraničená a úplná s oborom hodnôt a definičným oborom definovaným ako:

$$s : X \times X \mapsto I_s \subset R \tag{1}$$

kde $I_s$ je zhora ohraničená a $max_R I_S = S_{max}$. Funkcia podobnosti spĺňa aj nasledujúce vlastnosti:

1. reflexívnosť: $s(x, x) = s_{max}$
2. silná reflexívnosť: $s(x, y) = s_{max} \iff x = y$
3. symetrickosť: $s(x, y) = s(y, x)$
4. ohraničenosť: $s$ je ohraničená zdola ak: $\exists a \in R : s(x, y) \geq a, x, y \in X$
5. uzavretosť: $\exists x, y \in X : s(x, y) = s_{min}$
6. tranzitivita: $s(x, y) \geq \tau_s(s(x, z), s(z, y)), \forall x, y, z \in X$

Funkcia podobnosti očakáva na vstupe 2 objekty z množiny $X$ a výstupom je reálne číslo. Čím sa objekty viac podobajú, tým bude číslo väčšie.

### 4.1.2 Odlišnosť

Nech $X$ je neprázdna množina objektov s definovaným vzťahom rovnosti. Ak $d$ je funkcia odlišnosti, potom je $d$ zdola ohraničená a úplná s oborom hodnôt a definičným obrom definovaným ako:

$$d : X \times X \mapsto I_d \subset R \tag{2}$$

kde $I_d$ je zdola ohraničená a $max_R I_S = S_{min}$. Funkcia podobnosti spĺňa aj nasledujúce vlastnosti:

1. reflexívnosť: $d(x, x) = d_{min}$
2. silná reflexívnosť: $d(x, y) = d_{min} \iff x = y$

3. symetrickosť: $s(x, y) = s(y, x)$

4. ohraničenosť: $s$ je ohraničená zdola ak: $\exists a \in R : d(x, y) \leq a, x, y \in X$

5. uzavretosť: $\exists x, y \in X : d(x, y) = s_{max}$

6. tranzitivita: $d(x, y) \leq \tau_s(d(x, z), d(z, y)), \forall x, y, z \in X$

Funkcia odlišnosti očakáva na vstupe 2 objekty z množiny $X$ a výstupom je reálne číslo. Čím sú objekty odlišnejšie, tým bude číslo väčšie.

Okrem uvedených matematických vlastnosti by mali byť funkcie podobnosti a odlišnosti nezávislé od počtu pozorovaní a tried. Ďalej by hodnoty produkované týmito funkciami mali hladko stúpať od určitého minima pre podobné objekty po určité maximum pre odlišné objekty.

### 4.1.3 Vybrané miery podobnosti a odlišnosti

Predpokladajme, že všetky hodnoty vektorov $x_i$ a $x_j$ sú numerické, a že majú dimenziu $P$. Značenie $x_i(k), k \in 1, 2, ..., P$ označuje k-ty prvok vektora $x_i$.

1. **Euklidovská vzdialenosť** medzi $x_i$ a $x_j$ je definovaná ako

$$d_{euclidean}(x_i, x_j) = \sqrt{\sum_{k-1}^{P} [x_i(k) - x_j(k)]^2} \tag{3}$$

2. **Vážená euklidovská vzdialenosť** medzi $x_i$ a $x_j$ je definovaná ako

$$d_{euclidean}(x_i, x_j) = \sqrt{\sum_{k-1}^{P} W_k [x_i(k) - x_j(k)]^2} \tag{4}$$

   kde $W_k$ predstavuje váhu k-tého prvku.

3. **Gausian krenel** podobnosť medzi $x_i$ a $x_j$ je definovaná ako

$$s_{guasian}(x_i, x_j) = aexp(\frac{||x_i - x_j||^2}{2\sigma^2}) \tag{5}$$

   kde $\sigma > 0$ je rozptyl a $a$ je škálovacia konštanta. Výraz $||x_i - x_j||^2$ je euklidovská norma medzi $x_i$ a $x_j$.

4. **Kosínová podobnosť** medzi $x_i$ a $x_j$ je definovaná ako

$$s_{cosine}(x_i, x_j) = \frac{\sum_{k-1}^{P} x_i(k) x_j(k)}{||x_i||\ ||x_j||} = \frac{< x_i, x_j >}{||x_i||\ ||x_j||} \tag{6}$$

   kde $< x_i, x_j >$ je skalárny súčin a $||x_i||$ a $||x_j||$ je euklidovská norma.

## 4.2 Algoritmy pre konštrukciu sieti

Podľa autora knihy [2] môžeme algoritmy pre konštrukciu sietí rozdeliť do 3 skupín na základe typu využívanej vlastnosti:

1. **kvázi lokálna informácia** - tieto metódy využívajú čisto geometrické vlastnosti dát, ako je napríklad najkratšia vzdialenosť medzi objektmi. Medzi kvázi lokálne algoritmy sa radia
   - k-NN
   - $\epsilon$-radius
   - k-NNN + $\epsilon$-radius
   - NNN
   - LRNet
2. **informácia s kvázi ďalekým dosahom** - okrem geometrických vlastností využívajú napríklad aj informácie o najkratších cestách k objektom, ktoré po prevode nesusedia so skúmaným objektom. Do tejto skupiny sa radia:
   - b-matching network
   - linear neighborhood network
   - relaxed linear neighborhood network
   - formovanie siete pomcou zhlukovacích heuristik
3. **globálna informácia** - tieto metódy pri formovaní siete využívajú informácie o všetkých objektoch naraz. Príkladom globálnej informácie je distribúcia dát. Predstaviteľom tejto skupiny algoritmov je napríklad metóda založená na prekrývajúcich sa segmentoch histogramu [3], ktorá v tejto práci nie je bližšie popísaná.

Ďalšie delenie metód je založené na schopnosti prispôsobiť sa hustým respektíve riedkym oblastiam

1. **Statické metódy** - algoritmy v tejto kategórii sa nevedia prispôsobiť hustote dát, vďaka čomu pristupujú k dátam v riedkych aj hustých regiónoch uniformne. Typickí reprezentanti tejto kategórie sú $\epsilon - radius$ a $k$-NN
2. **Dynamické a adaptívne metódy** - algoritmy v tejto kategórii sa vedia prispôsobiť hustote dát. Do tejto kategórie patria napríklad:
   - k-NN + $\epsilon - radius$
   - NNN
   - LRNet

- b-matching network
- linear neighborhood network
- relaxed linear neighborhood network
- formovanie siete pomcou zhlukovacích heuristik

Ďalšiu veľkú skupinu metód uvedených v [2] tvoria metódy, ktoré prevádzajú časomerné údaje (angl *time-series*) na siete. Tento typ algoritmov je mimo záber tejto práce a preto nebude ďalej uvedený.

V následujúcej časti kapitoly je uvedených niekoľko metód, ktoré boli použité v práci ako aj algoritmy, ktoré neboli použité, ale patria do rovnakej kategórie.

### 4.2.1 k-NN

Trieda klasifikačných algoritmov *NN* (najbližších susedov) bola pôvodne určená pre klasifikáciu a regresiu vektorových dát, ale ich princíp môžeme tiež ľahko využiť na konštrukciu grafov. Vďaka svojej jednoduchosti a dobrým výsledkom patria algoritmy založené na najbližších susedoch medzi najobľúbenejšie algoritmy v oblasti klasifikácie a regresie.

Medzi typických predstaviteľov tejto triedy algoritmov patrí algoritmus *k-NN*. Algoritmus funguje na základe hľadania $k$ najbližších susedov. Najbližší susedia sú určení buď na základe podobnosti, alebo na základe vzdialenosti. V prípade použitia podobnosti sa vstupné záznamy zoradia zostupne, v prípade použitá vzdialenosti sa záznamy zoradia vzostupne. V ďalšom kroku sa použije prvých $k$ susedov a spočíta sa zastúpenie tried. Neoznačený záznam prevezme triedu s najväčším zastúpením.

Jeho najzákladnejšia varianta je algoritmus *1-NN*, kde neoznačený záznam preberá triedu najbližšieho suseda. Algoritmus *1-NN* garantuje maximálne dvojnásobok Bayesovskej miery chybovosti.[4]

Adaptácia algoritmu *k-NN* na konštrukciu grafu spočíva v tom, že neoznačený záznam nepreberá triedu z najväčším zastúpením, ale namiesto toho vytvorí hranu s $k$ najbližšími vrcholmi. Takto vytvorený graf je orientovaný.

### 4.2.2 $\epsilon$-radius

Algoritmus *$\epsilon$-radius* funguje na podobnom princípe ako algoritmus *k-NN*. V tomto prípade je trieda určená na základe $\epsilon$ okruhu, čo znamená, že sa určí reálne číslo $\epsilon$, pričom $\epsilon > 0$ a všetky záznamy, ktoré sú bližšie ako $\epsilon$, sú považované za susedov. Posledný krok je taký istý ako v prípade algoritmu *k-NN*. Neoznačený záznam preberá triedu s najväčším zastúpením, respektíve vytvorí hranu so všetkými susedmi. Takto vytvorený graf je neorientovaný.

### 4.2.3 Kombinácia k-NN a $\epsilon$-radius

V knihe [2] bola naznačená intuícia, že kombináciou techník *k-NN* a *$\epsilon$-radius* môžeme minimalizovať problémy, ktoré vyplývajú z ich statického charakteru.

Hlavným problémom algoritmu *k-NN* je, že ak je zvolené k príliš veľké, výsledkom sú hrany medzi vrcholmi, ktoré nie sú podobné.

Podobne pri algoritme *$\epsilon$-radius* môže nastať situácia, keď malé zvýšenie parametru $\epsilon$ spôsobí veľký nárast hustoty výsledného grafu.

Algoritmus založený na spojení tých algoritmov bol popísaný v [2] a funguje v dvoch krokoch. V prvom kroku sa každý vrchol prepojí s vrcholmi, ktoré sú v jeho *$\epsilon$-radius* okolí, a následne, ak je počet vrcholov, s ktorými sa vrchol prepojil v prvom kroku, menší ako $k$, použije sa *k-NN* tak, aby bol vrchol vo výsledku prepojený s $k$ vrcholmi.

### 4.2.4 NNN

Tento algoritmus má podobný princíp ako algoritmus *k-NN*. Algoritmus bol popísaný v [5] ako zhlukovací algoritmus. Podobne ako *k-NN*, *NNN* má len jeden vstupný parameter a tým je počet susedov $n$. Algoritmus zhlukovanie prebieha v troch krokoch.

1. každý vrchol vytvorí hranu s presne **n** najbližšími vrcholmi
2. odstránia sa hrany, ktoré nie sú symetrické, čiže ak existuje hrana $\mathbf{a} \rightarrow \mathbf{b}$, ale neexistuje hrana $\mathbf{b} \rightarrow \mathbf{a}$, hrana $\mathbf{a} \rightarrow \mathbf{b}$ je odstránená
3. identifikujú sa prekrývajúce sa kľuky a zlúčia sa do zhluku

V prípade využitia algoritmu na konštrukciu siete je krok 3. vynechaný.

### 4.2.5 LRNet

Metóda popísaná v [6] je založená na myšlienke, že objekty, ktoré sú v susedstve s ostatnými objektmi, sú dôležité. Pre popísanie algoritmu je dôležité uviesť niekoľko špeciálnych pojmov s ním spojenými.

Predpokladajme, že dataset $D = (O, s)$ kde $O$ je objekt datasetu a $s : O \times O \rightarrow R_{\geq 1}$ je funkcia podobnosti Susedstvo je formované na základe lokálnej reprezentatívnosti vrcholov. Pre výpočet **lokálnej reprezentatívnosti** objektov je najskôr potrebné vypočítať tzv. **základ x-reprezentatívnosti objektov**. Pre výpočet základu x-reprezentatívnosti objektu je potrebný lokálny stupeň objektu a lokálna dôležitosť objektu.
**Lokálny stupeň** $d(O_i)$ objektu je určený počtom objektov, ktorým je vrchol **x** susedom.
**Lokálna dôležitosť** $g(O_i)$ objektu je určená počtom objektov, ktorým je daný objekt najbližším susedom. Za najbližšieho suseda sa považuje objekt, ktorý je danému objektom najpodobnejší. Objekt môže mať viac najbližších susedov.

Ak **x-reprezentatívnosť** $r_x(O_i)$ definovaná následovne:

$$r_x(O_i) = \frac{g(O_i)}{log_x(1 + d(O_i))} \tag{7}$$

pričom $R_{>1}$ a $d(O_i)_{>1}$, potom je základ x-reprezentatívnosti $b(O_i)$ definovaný ako

$$b(O_i) = (1 + d(O_i))^{\frac{1}{g(O_i)}}) \tag{8}$$

Zo základu x-reprezentatívnosti je ďalej vypočítaná lokálna reprezentatívnosť objektov.

$$lr(O_i) = \frac{1}{b(O_i)} \text{ pre lokálne dôležité vrcholy} \tag{9}$$

$$lr(O_i) = 0 \text{ v ostatných prípadoch} \tag{10}$$

Reprezentatívny sused $k$ je určený ako:

$$k = ROUND(lr(O_i) \cdot d(O_i)) \tag{11}$$

a množina reprezentatívnych susedov objektu $O_i$ ako $K(O_i)$
Samotná konštrukcia siete prebieha v následujúcich krokoch:

1. vypočíta sa matica podobnosti $S$ pre dataset $D$
2. vypočíta sa lokálna reprezentatívnosť pre všetky objekty $O_i$
3. pre každý objekt $O_i$ sa vytvorí vrchol $v_i$
4. pre každú dvojicu vrcholov $v_i$, $v_j$ vznikne hrana $e_{ij}$ ak $O_j$ je najbližším susedom $O_i$ alebo ak $O_j$ je reprezentatívnym susedom $O_i$.

### 4.2.6 b-matching network

Na rozdiel od metódy k-NN, táto metóda zaisťuje presné spojenie každého vrcholu s $b$ inými vrcholmi. Pre neváženú maticu susedstva $A$, algoritmus b-matching využíva optimalizačný model (angl. *framework*), ktorý je definovaný následovne:

$$A^{min} \sum_{i,j \in V} A_{ij} D_{ij} \text{a platí } \sum_{j \in V} A_{ij} = bA_{ij} = 0A_{ij} = A_{ji} \tag{12}$$

kde $i, j \in V$, $D$ je matica odlišnosti a $D_{ij}$ je odlišnosť prvkov i-tého a j-tého prvku datasetu. Prvá podmienka zaisťuje, že každý vrchol je spojený s $b$ ostatnými, druhá podmienka vylučuje slučky a tretia podmienka zaisťuje symetrickosť.

Z definície optimalizačného algoritmu môžeme usúdiť, že až na poslednú podmienku, algoritmus prebieha rovnako ako $k - NN$. Posledná podmienka zapričiňuje, že sieť ním generovaná je pravidelná a vyvážená. [7]

### 4.2.7 Linear nighborhood network

Myšlienkou tejto metódy je vytvoriť sieť pomocou prekrývajúcich sa lineárnych vrstiev susedstiev (angl. *overlapped linear neighboorhod patches*), pričom váha hrán v každej vrstve susedstva je vypočítaná pomocou optimalizačnej techniky s názvom kvadratické programovanie. Na začiatku sú vytvorené susedstvá staticky, napríklad pomocou $k - NN$ alebo $\epsilon - radius$. Následne sú váhy hrán upravené pomocou lineárneho vkladania (angl. *linear embedning*), techniky pre redukciu dimenzie. Myšlienkou tejto techniky je využitie lokálneho susedstva objektu pri zostavovaní hrán. [8]

### 4.2.8 Relaxed linear neighborhood network

Táto metóda vylepšuje predchádzajúcu techniku s názvom linear nighborhood network. Vylepšenie spočíva v zostavovaní počiatočného susedstva s ohľadom na hustotu jeho okolia. Okrem toho je v tejto technike prehľadávané okolie susedstva, čím je odstránený vplyv ojedinelých pozorovaní a výsledná sieť je robustnejšia. [9]

### 4.2.9 Formovanie siete pomocou zhlukovacích heuristik

Ako naznačuje názov, ide o metódu založenú na technikách použitých pri zhlukovaní, konkrétne metódu *single-linkage*, v ktorej je za vzdialenosť zhlukov považovaná najkratšia vzdialenosť medzi dvoma objektmi rôznych zhlukov. Konštrukcia siete prebieha následovne:

1. pre každý objekt datasetu je vytvorený izolovaný vrchol, ktorý je zároveň zhlukom
2. vytvorí sa matica odlišnosti, na základe princípu metódy single-linkage
3. identifikujú sa dva najbližšie zhluky $G_1, G_2$
4. pre každý zhluk $G_1, G_2$ sa vypočíta priemerná odlišnosť a označí sa $d_1, d_2$
5. zvolí sa $k$ najpodobnejších hrán medzi zhlukmi $G_1$ a $G_2$ a vytvorí sa hrana medzi každým párom z $k$ hrán, pričom je zachovaná podmienka, že hrana je vytvorená len ak odlišnosť neprekročí stanovený prah $d_c = \lambda$
6. aktualizuje sa matica podobnosti. Hodnoty sú vypočítane zlúčením výsledkov z kroku č.5
7. ak je počet zhlukov väčší ako 1, prejde sa na krok č. 3. Tento krok zabezpečuje, že generovaná sieť je súvislá

Výsledné siete generované touto metódou sú súvislé a riedke, pričom dobre zachovávajú zhluky. [10]

# 5 Klasifikácia s využitím sieti

V kapitole 4.2 bolo naznačené, že vektorové dáta môžu byť reprezentované pomocou sietí. Siete prinášajú množstvo nových možností v oblasti strojového učenia. Tieto možnosti spočívajú predovšetkým v adaptácii algoritmov pracujúcich so sieťami na strojové učenie. V knihe [2] bolo naznačené, že na rozdiel od oblasti učenia sa bez učiteľa a učenia sa s čiastočnou pomocou učiteľa, v ktorých vzniklo väčšie množstvo algoritmov, oblasť učenia sa s učiteľom počtom algoritmov zaostáva.

Okrem skupiny metód pracujúcich so sieťami, ktoré boli vytvorené z vektorových dát, môžeme identifikovať ďalšiu skupinu metód zameraných na tzv. relačnú klasifikáciu. Tento typ klasifikácie sa zaraďuje do kategórie strojového učenia s učiteľom a pracuje so špecifickým typom dát. Tieto dáta porušujú predpoklad klasifikácie, ktorý hovorí, že trieda objektu by mala závisieť len na jeho atribútoch, a to tým, že ich trieda závisí aj na triede okolitých objektov. Technika relačnej klasifikácie bola použitá na identifikáciu "ciest"v génoch a na interakciu proteínov [11], na klasifikáciu prepojenia vedeckých prác [12] a na predikciu hrán. Predikcia hrán spočíva v odhade, či medzi dvoma vrcholmi v budúcnosti vznikne hrana. Ďalším využitím relačnej klasifikácie je hľadanie podgrafov spájajúcich dva vrcholy, ktoré najlepšie popisujú ich vzájomný vzťah. V tejto oblasti vznikla práca založená na zákonoch elektrických okruhov [13].

Ďalšiu skupinu tvoria metódy založené na princípe kolektívnej zhody. Kolektívna klasifikácia je prístup, v ktorom je trieda prepojených vrcholov určená zjednotene (angl. *jointly*). Na dosiahnutie kolektívnej klasifikácie sú použité algoritmy kolektívnej dedukcie (angl. *collective inference*). Kolektívna klasifikácia bola rozsiahlo skúmaná v [14]. Výsledky tejto štúdie ukázali schopnosť algoritmov kolektívnej klasifikácie výrazne zlepšiť presnosť bežných klasifikačných algoritmov, avšak v prípade nepresnej propagácie v prvom kroku klasifikácie výsledky naopak zhoršia. V porovnaní s bežnými algoritmami pre relačnú klasifikáciu , pri ktorých má na výsledok klasifikácie vrcholu vplyv predovšetkým vrchol s ním spojený, algoritmy kolektívnej dedukcie využívajú okrem samotného objektu (vrcholu) aj informácie získaných z objektov, ktoré sú s týmto objektom prepojené a kolektívne dedukujú výsledné zaradenie jedného alebo viacerých vrcholov naraz.

## 5.1 Average Link Weight

Tento algoritmus je inšpirovaný mierou uvedenou v [6] a je založený na priemernom súčte váh hrán. Testovací vrchol preberá triedu na základe súčtu priemernej váhy susedných vrcholov. Algoritmus má jeden parameter, ktorým je počet hrán použitých na klasifikáciu. Hrany môžu byť nastavené ručne alebo na základe určitej heuristiky. Medzi základné prístupy patrí použitie štatistických údajov spojených so stupňom vrcholu, ako napríklad použitie priemerného stupňa, mediánu alebo kvartilov.

### 5.1.1 Fáza trénovania

Vo fáze trénovania je zostrojený vážený a neorientovaný graf bez slučiek. Vrcholy reprezentujú označkované objekty datasetu a hrany medzi nim sú zostrojené a vážené pomocou funkcie podobnosti a techniky formovania siete popísanej v 4.2.

### 5.1.2 Fáza testovania

Na začiatku algoritmu je staticky určený počet vrcholov $k$, ktoré sú použité na určenie triedy. Počet je zvolený ručne alebo na základe určitej heuristiky. V ďalšom kroku sa postupne vkladajú testovacie vrcholy do tréningovej siete tak, že vrchol je spojený s $k$ najbližšími susedmi. Následne je spočítaný súčet váh hrán s každou triedou a trieda, ktorá má mať najväčší súčet, je určená ako výsledok klasifikácie. V poslednom kroku sa testovací vrchol opäť odoberie zo siete. Celková presnosť klasifikácie je určená ako priemerná úspešnosť testovacích vrcholov.

## 5.2 Ease Of Access

Tento klasifikátor bol popísaný v [15] a klasifikácia sa opiera o heuristiku nazývanú *jednoduchosť prístupu* využívanú v prostredí sietí. Meranie jednoduchosti prístupu sa opiera o tzv. ustálený vektor Markovovho reťazca.

**Markovov reťazec** popisuje náhodný proces, v ktorom platí, že prechod následujúceho stavu závisí iba na súčasnom stave. Vďaka tejto vlastnosti sa dá proces jednoducho znázorniť pomocno stavového diagramu, pričom jednotlivé prechody reprezentujú pravdepodobnosť zvolenia prechodu. Pre reťazce s väčším počtom stavov je vhodnejšia reprezentácia pomocou matice pravdepodobnosti $A$.

**Ustálený vektor** Markovovho reťazca $v^*$ existuje v prípade, že reťazec je neredukovateľný a aperiodický. Markovov reťazec je **neredukovateľný** v prípade, že existuje cesta medzi všetkým stavmi reťazca. Neredukovateľný Markovov reťazec je periodický ak exituje $t \in 2, 3, 4, ...$ také, že reťazec sa môže nachádzať v určitom stave len v čase $t, 2t, 3t, 4t, ...$ čiže s periódou $t$. Ak takýto stav v reťazci neexistuje, reťazec je **aperiodický**.

Pre každý Markovov reťazec s $k$ stavmi a v čase $t$ existuje vektor $v_t = (v_t[1], ..., v_t[k])$ obsahujúci distribúcie pravdepodobnosti.
Ak je Markovov reťazec neredukovateľný a aperiodický, vektor pravdepodobnosti $v$ sa po dostatočne veľkom počte krokov $t$ bude približovať vektoru $v^*$

V procese klasifikácie sú vrcholy grafu považované za stavy a vážené hrany medzi vrcholmi sú považované za prechody, pričom váha hrany je reprezentovaná podobnosťou objektov.
Klasifikácia datasetu $D$ rozdeleného na podmnožinu $L$ označkovaných objektov a podmnožinu $U$ neoznačkovaných objektov je rozdelená do dvoch fáz.

### 5.2.1 Fáza trénovania

Podobne ako pri predchádzajúcom algoritme, je vo fáze trénovania zostrojený vážený a neorientovaný graf bez slučiek. Vrcholy reprezentujú označkované objekty datasetu a hrany medzi nim sú zostrojené a vážené pomocou funkcie podobnosti a techniky formovania siete popísanej v 4.2.

### 5.2.2 Fáza testovania

Klasifikácia neoznačkovaného objektu $x$ prebieha v následujúcich krokoch:

1. vytvorí sa vektor podobnosti $s = [s_1, s_2, ..., s_L]$, pričom prvky $s_i$ sú podobnosti objektu $x$ s označkovanými objektmi, ktoré sú vypočítane pomocou funkcie podobnosti.

2. ďalej je vytvorená asymetrická a vážená matica podobnosti s $L$ vrcholmi.

$$\hat{A} = A + \epsilon\hat{S} \tag{13}$$

   kde$A$ a $\hat{A}$ a sú originálna a šumová matica susednosti, $\epsilon$ je parameter algoritmu $\epsilon > 0$ a $\hat{S}$ je nasledujúca matica $L \times L$:

$$\hat{S} = \begin{bmatrix} S_{(1)} \\ S_{(2)} \\ \vdots \\ S_{(L)} \end{bmatrix} \tag{14}$$

   kde $S_{(i)}$ je riadkový vektor $L \times 1$, ktorého prvky sú všetky $s_i$.

3. Šumová matica $\hat{A}$, ktorá reprezentuje tréningovú sieť, je nasledovne transformovaná na maticu pravdepodobností $P$"

$$P_{ij} = \hat{A_{ij}} / \sum_{j \in V} \hat{A_{ij}} \tag{15}$$

4. vypočíta sa ustálený vektor $v$, ktorý reprezentuje s akou pravdepodobnosťou bude mať neoznačkovaný vrchol $x$ rovnakú triedu ako vrchol na pozícii $v[k] : k \in 0, 1, .., L$.

5. v poslednom kroku sa z $T$ vrcholov určí trieda s najväčším zastúpením ako je výsledok algoritmu

Vďaka faktu, že obidve metódy rozdeľujú algoritmus na fázu trénovania a fázu testovania, sú vhodné na testovanie pomocou metód *k-násobná krížová validácia* alebo *leave one out* uvedených v 2.2. Obidve metódy sú v práci implementované a použité v experimentoch v kombinácii s validáciou pomocou *k-násobnej krížovej validácie*

# 6 Implementácia

Za účelom overenia teórie a automatizácie experimentov vzniklo 5 knižníc a jedna aplikácia na použitie týchto knižníc. Pri implementácii knižníc bol kladený dôraz predovšetkým na zapuzdrenie a rozšíriteľnosť, zatiaľ čo hlavným cieľom aplikácie bol jednoduchý prístup k funkcionalite knižníc a vizualizácia výsledkov produkovaných týmito knižnicami. Okrem základnej funkcionality, ktorá je nutná pre prevod a klasifikáciu, a ktorá bola implementovaná pred vykonaním experimentov, boli počas vykonávania experimentov identifikované a čiastočné implementované ďalšie prípady použitia.

## 6.1 Structures.Graph

Obsahuje základné štruktúry pre prácu s grafom.

### 6.1.1 Trieda Graph

Trieda obsahuje zoznam vrcholov reprezentovaných triedou **Node** a zoznam hrán reprezentovaných triedou **Link**.
Okrem základných metód pre manipuláciu s vrcholmi a hranami obsahuje jednu factory metódu **FromMatrix()**, ktorá vytvára graf z matice súslednosti.

### 6.1.2 Trieda Node

reprezentujúca vrchol grafu.

### 6.1.3 Trieda Link

reprezentuje hranu medzi dvoma vrcholmi.

## 6.2 Structures.Vector

Táto knižnica poskytuje rozhranie pre načítanie csv súboru a jeho reprezentáciu vo forme datasetu.

### 6.2.1 Trieda Dataset

zaobaľuje hlavne vlastnosti datasetu, ako sú napríklad zoznam tried, zoznam, atribútov a zoznam objektov typu **Datarow**.
Okrem prístupu k objektom obsahuje nasledujúce verejné metódy:

- **GetNormalizedEntries()** táto metóda vracia *kópiu* objektov s normalizovanými hodnotami atribútov. Normalizácia je realizovaná metódou min-max:

$$x_i^j = \frac{x_i^j - min_j}{max_j - min_j} \tag{16}$$

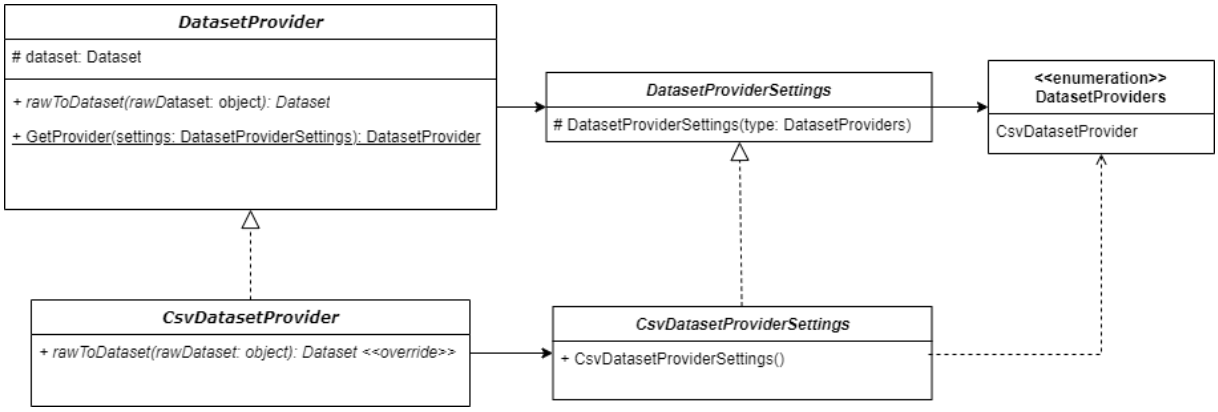
- **SplitIntoKFolds(entries, k)** statická metóda, ktorá náhodne rozdelí zoznam objektov do k skupín, pričom každá skupina má rovnomerné zastúpenie tried.

### 6.2.2 Trieda Datarow

reprezentuje jeden objekt datasetu.

### 6.2.3 Spracovanie Datasetu

Za účelom spracovania datasetu vznikla generická kolekcia tried umožňujúca jednotné načítanie datasetu z rôznych zdrojov. Rozhranie definované touto kolekciou bolo implementované v podobe načítania datasetu zo súboru s príponou *.csv*.



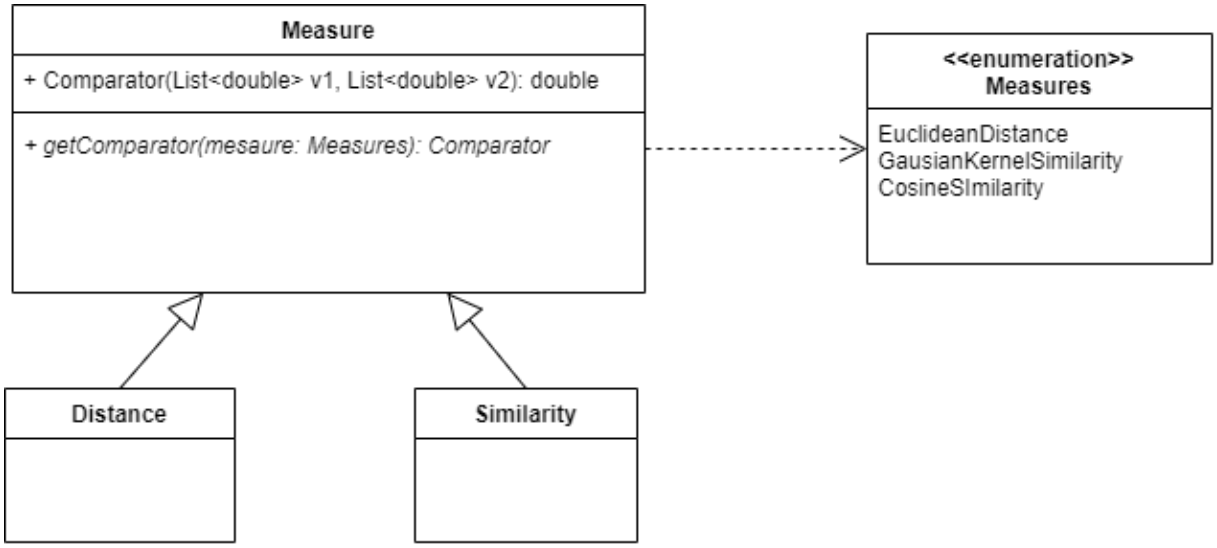

Obr. 1: Triedny diagram: načítanie dát zo súbor s príponou .csv

Samotné spracovanie datasetu prebieha v 2 krokoch:

1. vytvorenie inštancie potomka triedy **DatasetProvider**
2. spracovanie datasetu zavolaním abstraktnej metódy **rawToDataset()**

## 6.3 Analytics.Measures

Táto knižnica poskytuje metódy nutné na porovnanie dvoch objektov datasetu.




Obr. 2: Triedny diagram: kľúčové triedy pre výpočet podobnosti a vzdialenosti

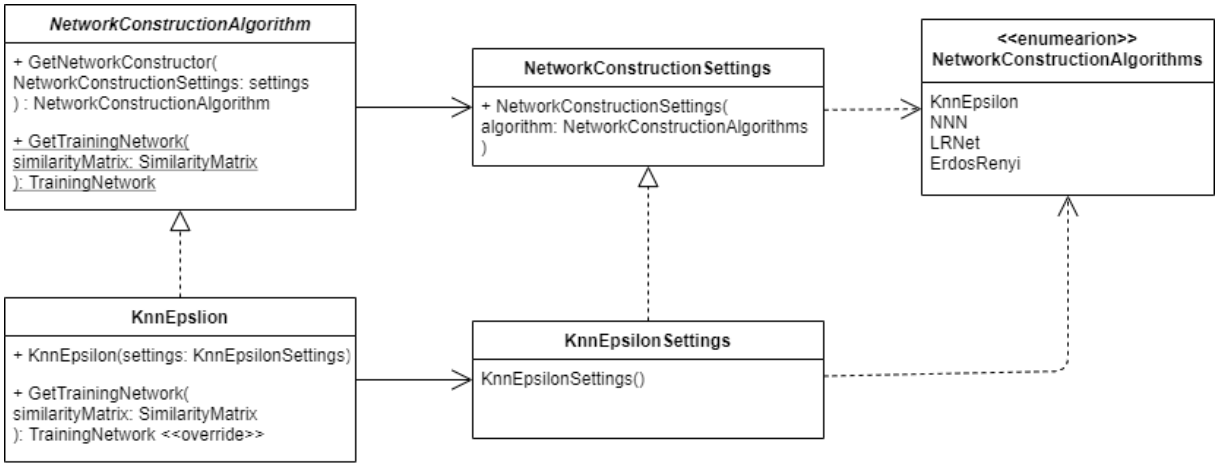
Knižnica využíva konštrukciu jazyka csharp s názvom delegát [16]. Delegát je dátová štruktúra reprezentujúca referenciu na funkciu. Delegát ma striktne určený počet a typ parametrov a návratovú hodnotu.

Delegát s názvom **Comparator** očakáva na vstupe 2 vektory rovnakej dimenzie. Výstupom je podobnosť alebo vzdialenosť. Podobnosť, ktorá bude použitá, je určená parametrom metódy **getComparator()**.

## 6.4 Analytics.NetworkConstruction

Pre transformáciu datasetu z vektorovej formy, do formy grafu boli implementované 4 algoritmy:

- k-NN + $\epsilon-$radius
- LRNet
- NNN
- ErdosRenyi




Obr. 3: Triedny diagram: prevod vektorových dát na sieť pomocou algoritmu kNN + $\epsilon - radius$

Podobne ako pri spracovaní datasetu, generické rozhranie definuje továrenskú metódu **GetNetworkConstructor()** na vytvorenie inštancie triedy zodpovednej za transformáciu. Nad vytvorenou inštanciou je možné zavolať abstraktnú metódu **GetTrainingNetwork()**, ktorej výstupom je inštancia triedy TrainingNetwork.

Trieda **TrainingNetwork** je abstrakcia nad štandardnou implementáciu viacrozmerného poľa jazyka C#. Táto trieda obsahuje okrem siete uloženej vo forme matici susednosti aj referenciu na *Comparator*, funkciu použitú na zostavenie matice susednosti. Ďalšou dôležitou súčasťou je mapovanie Identifikátora objektu datasetu na záznam v matici susednosti. Mapovanie je realizované dátovou štruktúrou *Dictionary<int, Datarow>*, kde kľúč je riadok matice a hodnota je zodpovedajúci objekt datasetu. Mapovanie je dôležité predovšetkým, ak je graf produkovaný touto knižnicou použitý pri kros-validácii, kde je z datasetu vygenerovaných niekoľko sietí z rôznych podmnožín objektov, ktoré sú náhodne zoradené. Alternatívne riešenie implementácie by mohlo použiť referencie na objekty namiesto ich identifikátorov. Toto riešenie prináša zníženie potrebného výkonu na operácie s maticami a preto nebolo implementované.

Algoritmus KNN + $\epsilon-$radius pri nastavení parametru k = 0 funguje ako základný $\epsilon-$radius algoritmus, rovnako ako pri nastavení $\epsilon = 0$ algoritmus funguje ako algoritmus k-NN.

Algoritmus ErdosRenyi generuje náhodný model siete [17] a bol implementovaný ako referenčný algoritmus pre porovnanie výsledkov s ostatnými metódami.

V priebehu experimentov bola táto knižnica rozšírená o kolekciu tried, ktoré slúžia k hľadaniu parametrov algoritmov pre požadovanú hustotu siete. Vzhľadom na fakt, že niektoré algoritmy vyžadujú nastavenie viacerých parametrov, existuje viac možností ako danú hustotu dosiahnuť. Hľadanie najlepšej siete pre danú hustotu však reprezentuje problém viacúčelovej optimalizácie, ktorá presahuje zameranie tejto práce a preto boli využité jednoduchšie techniky na základe náhody. Bližšie informácie o použití tried je popísane v experimente 7.3.

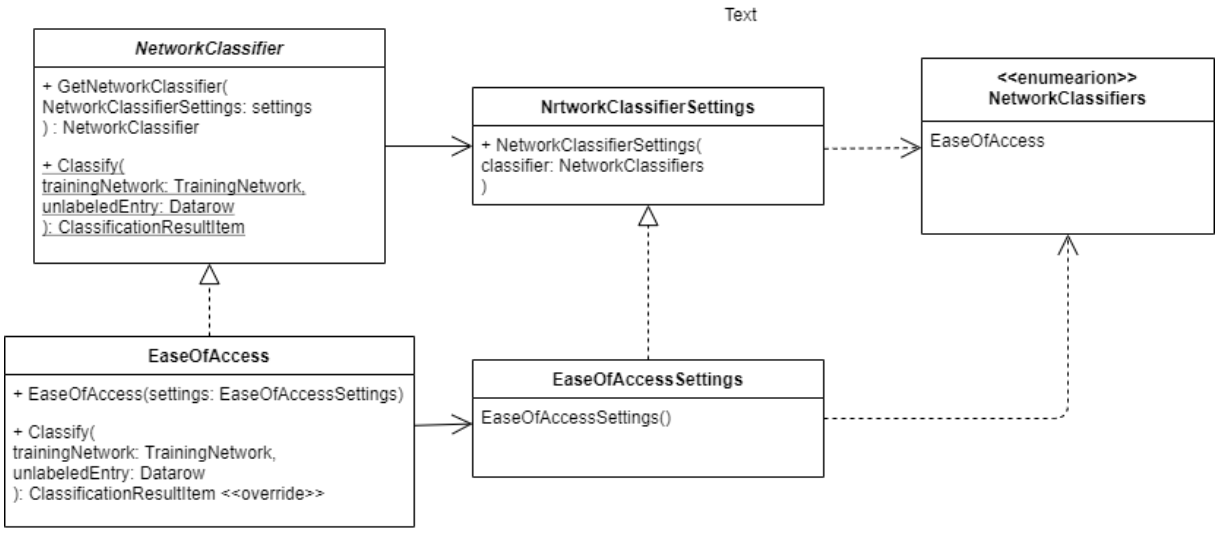
Triedy vytvorené na hľadanie parametrov pre požadovanú hustotu môžu byť použité aj na exploračnú analýzu, a to vďaka implementácií, ktorá uchováva pre každý krok hľadania dôležité

informácie o sieti. Tieto informácie môžu byť použité pri ladení sieti a klasifikačných algoritmov.

Štruktúra dedičnosti tried na hľadanie hustoty je podobná štruktúre tried na samotný prevod. Základom je abstraktná trieda **DensityDiscoverer**, ktorá definuje jednu abstraktnú metódu s názvom **Find()** a parametrom reprezentovaným abstraktnou triedou **DensityDiscovererSettings**. Výstupom je pole objektov typu **DensityMemento**. Pre každý algoritmus sú vytvorene 3 triedy, ktoré dedia z vyššie uvedených tried a obsahujú implementáciu špecifickú pre daný algoritmus.

## 6.5 Analytics.NetworkClassification

Podobne ako pri spracovaní datasetu alebo konštrukcii siete, najskôr bola vytvorená generická kolekcia tried definujúca rozhranie a následne bol implementovaný algoritmus Ease of Access.




Obr. 4: Triedny diagram: klasifikácia pomocou algoritmu Ease of Access

Klasifikátor očakáva na vstupe sieť, ktorá bola zostrojená z označkovaných objektov datasetu, a neoznačkovaný objekt. Výstupom algoritmu je inštancia triedy ClassificationResultItem.
**ClassificationResultItem** obsahuje výsledok klasifikácie, referenciu na neoznačkovaný objekt a kolekciu objektov, z ktorých bola výsledná trieda propagovaná.

Knižnica ďalej obsahuje triedu **Classification**. Verejné rozhranie tejto triedy je tvorené 2 metódami.
Metóda **ClassifyOne** očakáva na vstupe id objektu, nad ktorým má previesť klasifikáciu. Výstupom tejto metódy je ClassificationResultItem.
Metóda **ClassifiyMany** je bezparametrická a slúži na k-násobnú kros-validáciu tréningovej siete.
Kros-validácia prebieha v $n$ iteráciách. V každej iterácii je dataset rozdelený do $k$ podmnožín. Prvky podmnožiny sú volené náhodne, pričom je zachovaná rovnomerná distribúcia tried, ktorá

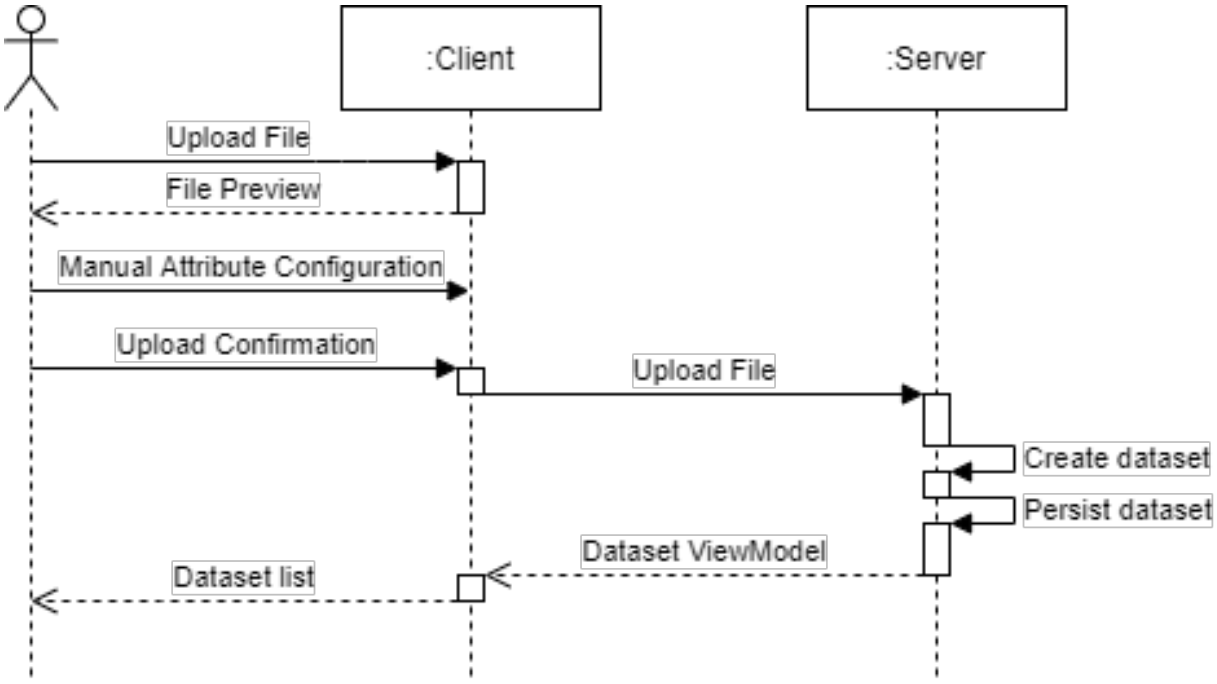
je dosiahnutá algoritmom round-robin.
Výstupom tejto metódy je **ClassificationReport**.

## 6.6 Aplikácia

Aplikácia je implementovaná v jazyku .NET Core a opiera sa o architektúru client-server. Na serverovej časti bolo vytvorené WEB API, na ktoré bol použitý framework ASP.NET Core 3.1. Klient bol implementovaný ako Angular 9 aplikácia.

### 6.6.1 Spracovanie datasetu

Prvým krokom klasifikácie je voľba datasetu. Za týmto účelom bol vytvorený formulár na nahranie datasetu vo forme súboru s príponou *.csv*.

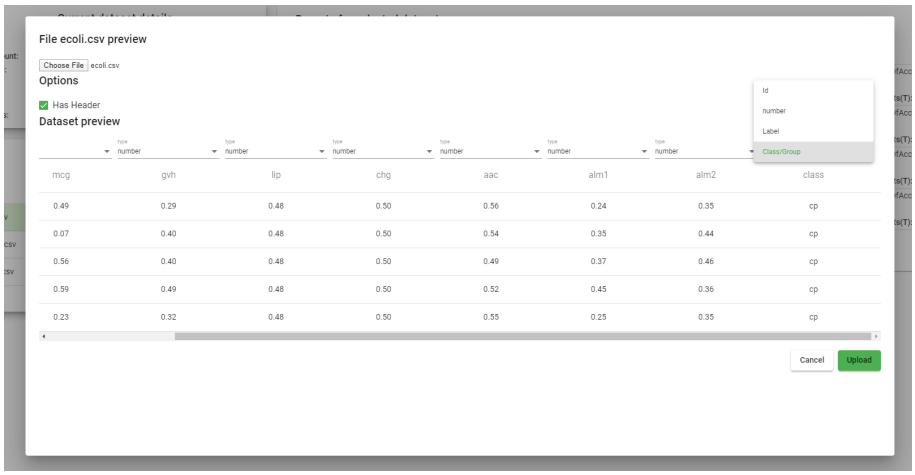



Obr. 5: Sekvenčný diagram: nahranie datasetu

Po otvorení formulára užívateľ najskôr zvolí cestu k súboru, ktorý chce nahrať. Klient následne spracuje súbor a zobrazí náhľad dát. Súčasťou spracovania súboru je okrem detekcie základnej štruktúry súboru aj detekcia typu niektorých atribútov. Nasledujúce typy atribútov sú detegované automaticky na základe názvu stĺpca:

- **id** - identifikátor objektu
- **label/name** - štítok objektu
- **class/group** - trieda objektu

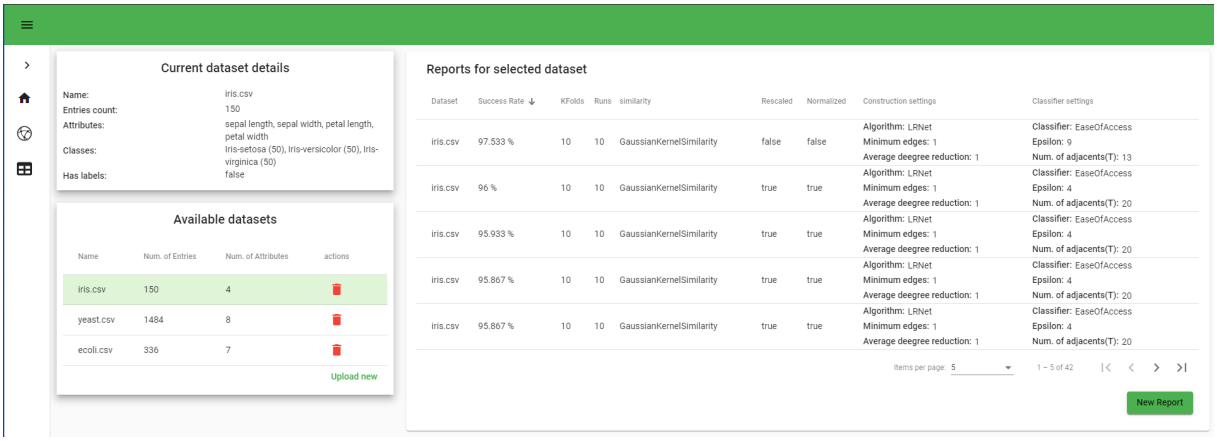
Ak názov stĺpca nezodpovedá vyššie uvedeným typom, použije sa prednastavený typ **attribute**, identifikujúci atribút objektu, ktorý sa ďalej využíva na výpočet podobnosti alebo vzdialenosti. Ak súbor neobsahuje hlavičku s názvami stĺpcov, názvy sú vygenerované vo formáte $a = a0, a1, ..., an$. V ďalšom kroku môže užívateľ manuálne zmeniť detegované typy atribútov a potvrdiť spracovanie. Po potvrdení spracovania je súbor spolu s metadátami odoslaný na server, kde je z neho vytvorený dataset, ktorý je následne uložený v databáze. Po spracovaní a uložení datasetu je vytvorený náhľad datasetu, ktorý je odoslaný ako odpoveď klientovi a zobrazený užívateľovi.




Obr. 6: Ukážka aplikácie: formulár pre nahranie datasetu

### 6.6.2 Domovská stránka

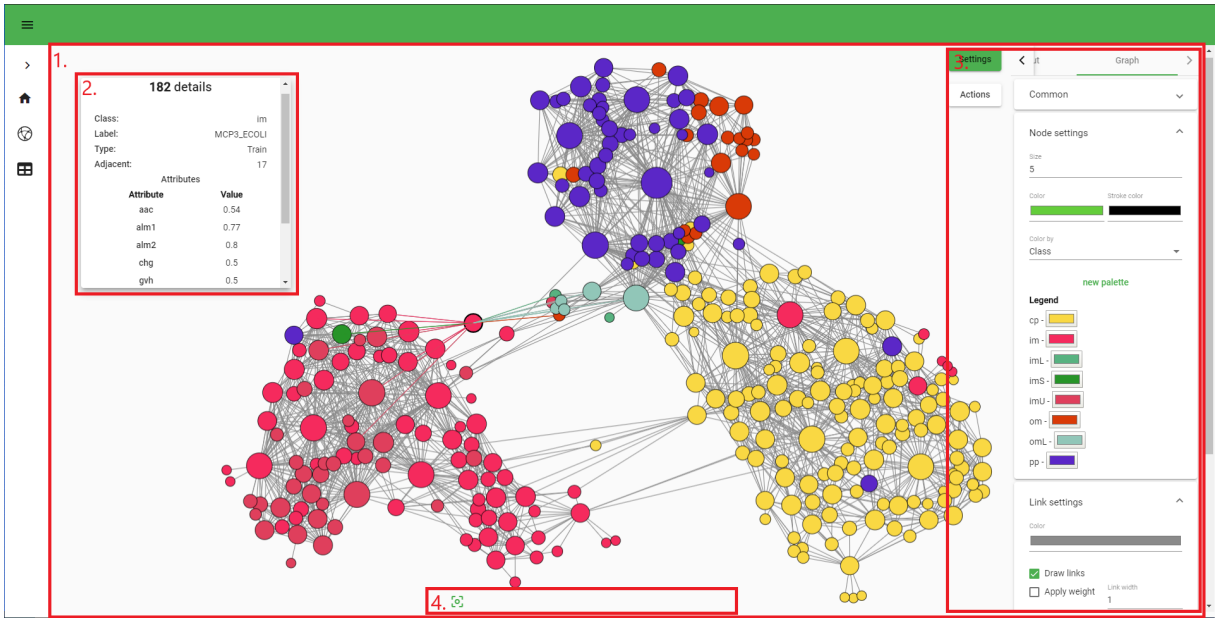
Po načítaní aplikácie je užívateľovi zobrazený zoznam uložených datasetov. Po kliknutí na dataset sa zobrazí jeho detail a uložené výsledky klasifikácii nad týmto datasetom.




Obr. 7: Ukážka aplikácie: domovská stránka

### 6.6.3 Vizualizácia siete

Pre potreby aplikácie vznikol modul obsahujúci interaktívny model grafu. Vzhľadom k špecifickým požiadavkám aplikácie a nedostatočnej podpore už existujúcich riešení pre interaktívnu prácu s grafom, bola zvolená knižnica *d3.js*, ktorá poskytuje množstvo funkcií na implementáciu požadovanej funkcionality.




Obr. 8: Ukážka aplikácie: rozhranie pre vizualizáciu siete

Na obrázku 8 môžeme vidieť 4 zvýraznené komponenty.

1. **Graf** - hlavný komponent obsahujúci sieť vykreslenú pomocou HTML elementu s názvom *<canvas>*. Medzi implementované prvky interakcie patrí:
   - **zoom** - približovanie a odďaľovanie grafu
   - **pan** - posúvanie grafu
   - **drag** - presúvanie vrcholov
   - **select** - zvolenie a zvýraznenie vrcholu
2. **Detail vrcholu** - komponent zobrazujúci detaily zvoleného vrcholu
3. **Postranne menu** - komponent obsahuje 2 logické celky.
   (a) **settings** - menu nastavenia grafu, ktoré je ďalej rozdelené na
      i. nastavenia vizuálnych vlastnosti grafu
      ii. nastavenia rozloženia grafu

(b) **actions** - nepovinné menu obsahujúce funkcionalitu špecifickú na použitie grafu. Táto konštrukcia umožňuje dodať funkcionalitu na rôzne užívateľské scenáre, ako napríklad ladenie grafu, kde menu **actions** obsahuje komponent na generovanie tréningovej siete.

4. **Menu s rýchlymi akciami** - tento komponent slúži ako kontajner na ďalšiu interakciu s grafom a obsahuje jednu funkciu, ktorá resetuje pohľad a priblíženie na prednastavené hodnoty.

   Okrem komponentov slúžiacich na vizualizáciu siete modul obsahuje aj komponent na rozloženie grafu realizovaný pomocou služby. Rozloženie grafu, tiež známe ako **layout**, prebieha kvôli výpočtovej náročnosti v samostatnom vlákne a nové pozície vrcholov sú priebežne zasielané na hlavné vlákno, pomocou ktorého sú vykreslené. Pokročilé nastavenia algoritmu na generovanie rozloženia sú prístupné v postrannom menu.

### 6.6.4 Ladenie siete

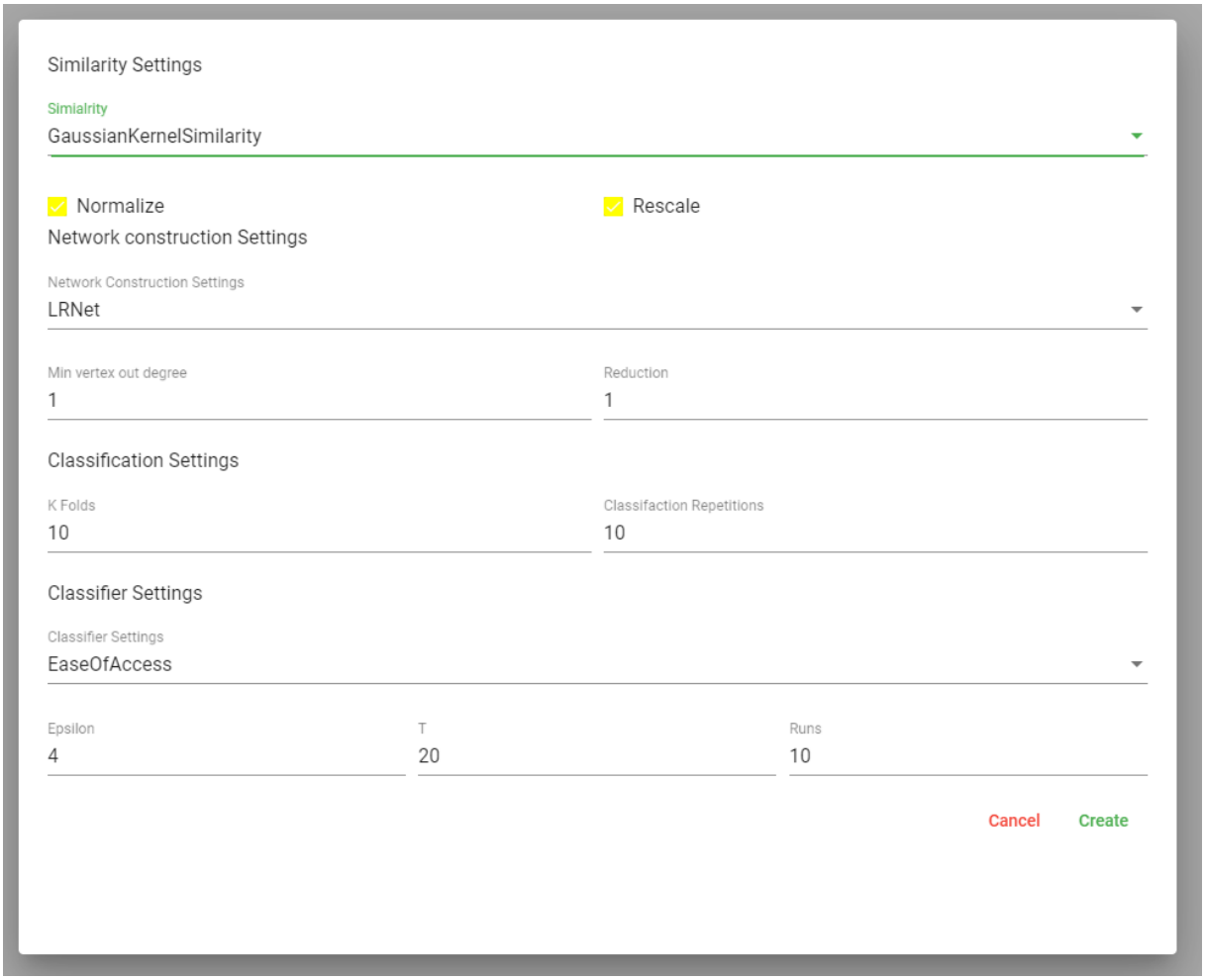
S využitím modulu na vizualizáciu a interakciu so sieťou 6.6.3 bol implementovaný komponent na ladenie siete. Tento komponent umožňuje rýchly náhľad na dataset v podobe siete a jej následne ladenie na použitie na klasifikáciu.

### 6.6.5 Ladenie klasifikácie

Ladenie samotnej klasifikácie prebieha pomocou k-násobnej krížovej validácie. Užívateľ vyplní formulár, ktorý je odoslaný na server, kde prebehne samotná klasifikácia. Nastavenie klasifikácie môžeme rozdeliť do 4 krokov.

1. nastavenie podobnosti
2. nastavenie algoritmu pre konštrukciu siete
3. všeobecné nastavenie klasifikácie
4. nastavenie klasifikačného algoritmu

Na obrázku 9 môžeme vidieť formulár na nastavenie klasifikácie, pričom vstupy na nastavenie algoritmu na konštrukciu siete a nastavenie klasifikačného algoritmu sú zobrazené dynamicky na základe zvoleného algoritmu

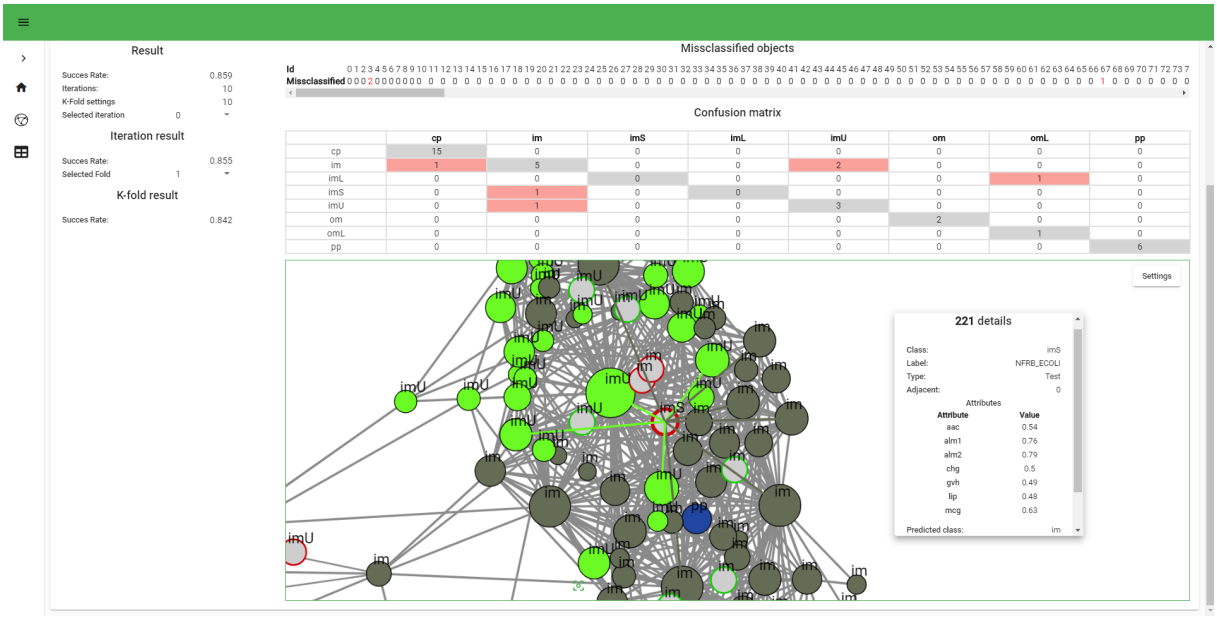



Obr. 9: Ukážka aplikácie: formulár pre nastavenie klasifikácie

Po dokončení klasifikácie je užívateľovi odoslaný výsledok, ktorý je následne vizualizovaný. Komponent na vizualizáciu výsledku je rozdelený na následujúce časti:

1. **Použité nastavenie**
2. **Výsledok klasifikácie** - okrem výsledku obsahuje rozhranie na zmenu aktuálne zobrazených detailov.
3. **Nesprávne klasifikované objekty** - tabuľka obsahujúca počet nesprávnych klasifikácii pre každý objekt.
4. **Confusion matrix** - zobrazuje maticu, kde stĺpce zobrazujú reálnu triedu objektu a riadky predikovanú.
5. **Použitá sieť** - interaktívny graf s využitím modulu na vizualizáciu a interakciu so sieťou 6.6.3. Vrcholy testovacích objektov sú v grafe zobrazené šedou farbou. Ich okraje sú zelené ak bola predikovaná trieda správna alebo črevné ak bola predikovaná trieda nesprávna. Po

kliknutí na testovací objekt sú zobrazené hrany medzi s vrcholmi, ktoré boli použité pre propagáciu triedy.




Obr. 10: Ukážka aplikácie: rozhranie pre zobrazenie výsledku klasifikácie

# 7 Experimenty

Experimenty vykonané v tejto práci sa zameriavajú na porovnanie a vyhodnotenie klasifikácie pomocou sietí. Na experimenty boli zvolené algoritmy *Ease of Acces* a *Averrage Link Weigth*.

Na prevod vektorových dát na sieť boli použité algoritmy $kNN + \epsilon - radius$, *NNN* z *LRNet* a referenčný algoritmus pre generovanie náhodných sietí *Erdos-Renyi*. Voľba správneho algoritmu na vytvorenie siete zohráva pri klasifikácii pomocou siete dôležitú úlohu, a preto je v experimentoch taktiež detailne skúmaná.

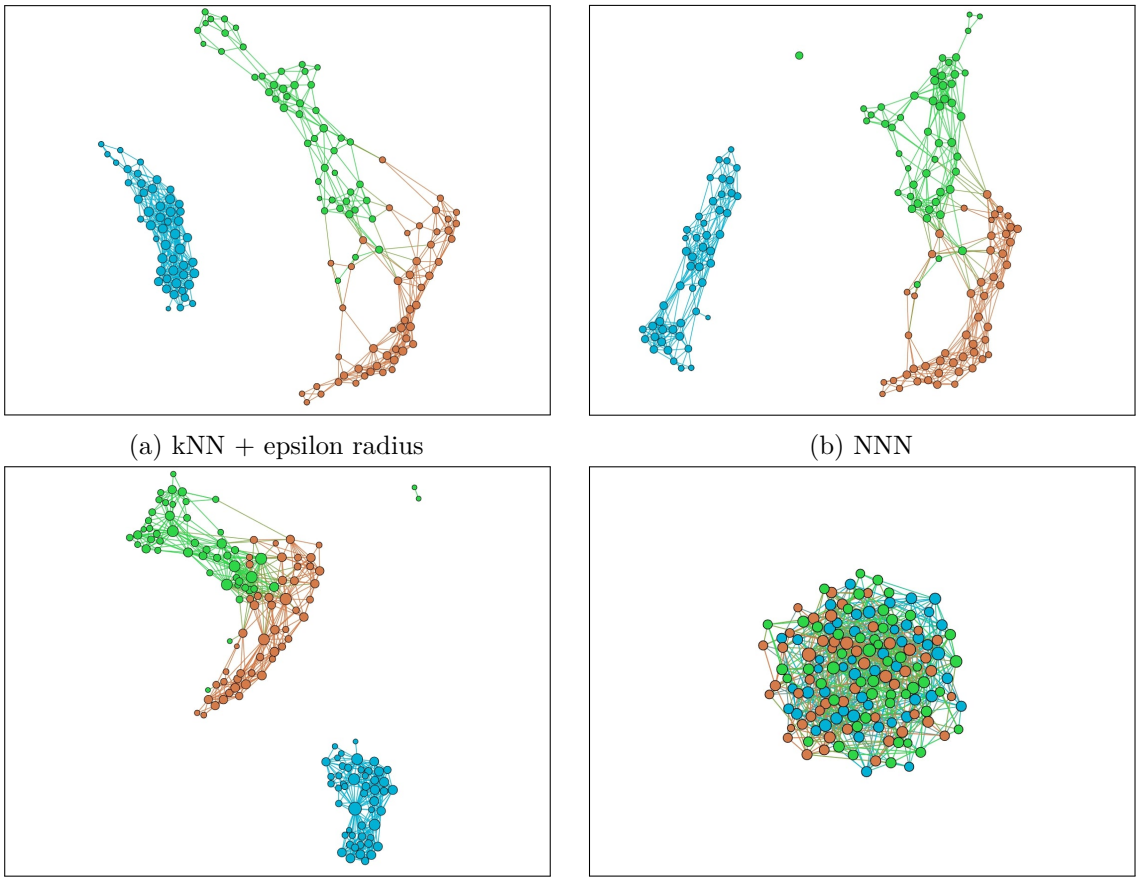
Na začiatku tejto kapitoly sú uvedené datasety, ktoré boli použité v experimentoch. Datasety boli zvolené z typických datasetov v oblasti klasifikácie, pričom boli preferované datasety s nižším počtom vrcholov, čo umožňuje ľahšiu vizualizáciu a nižšiu výpočtovú náročnosť. Ku každému datasetu boli vytvorené siete pomocou implementovaných algoritmov. Siete boli vytvorené s rovnakou hustotou na lepšie vizuálne porovnanie.

## 7.1 Použité datasety

### 7.1.1 Iris dataset

Ide asi o najznámejší dataset v oblasti klasifikácie. Vďaka malému počtu pozorovaní rozdelených do 3 rovnako veľkých skupín je tento dataset vhodný na rýchle testovanie algoritmov, vizualizáciu a testovanie predpokladov.

Dataset obsahuje 150 záznamov a 4 atribúty. Záznamy sú rovnomerne rozdelené do troch tried. Jednotlivé záznamy predstavujú kvety z rodu Iris a atribúty predstavujú ich proporcie. Na základe proporcií je rastlina zaradená do jednej z troch odrôd.[18]



(a) kNN + epsilon radius

(b) NNN

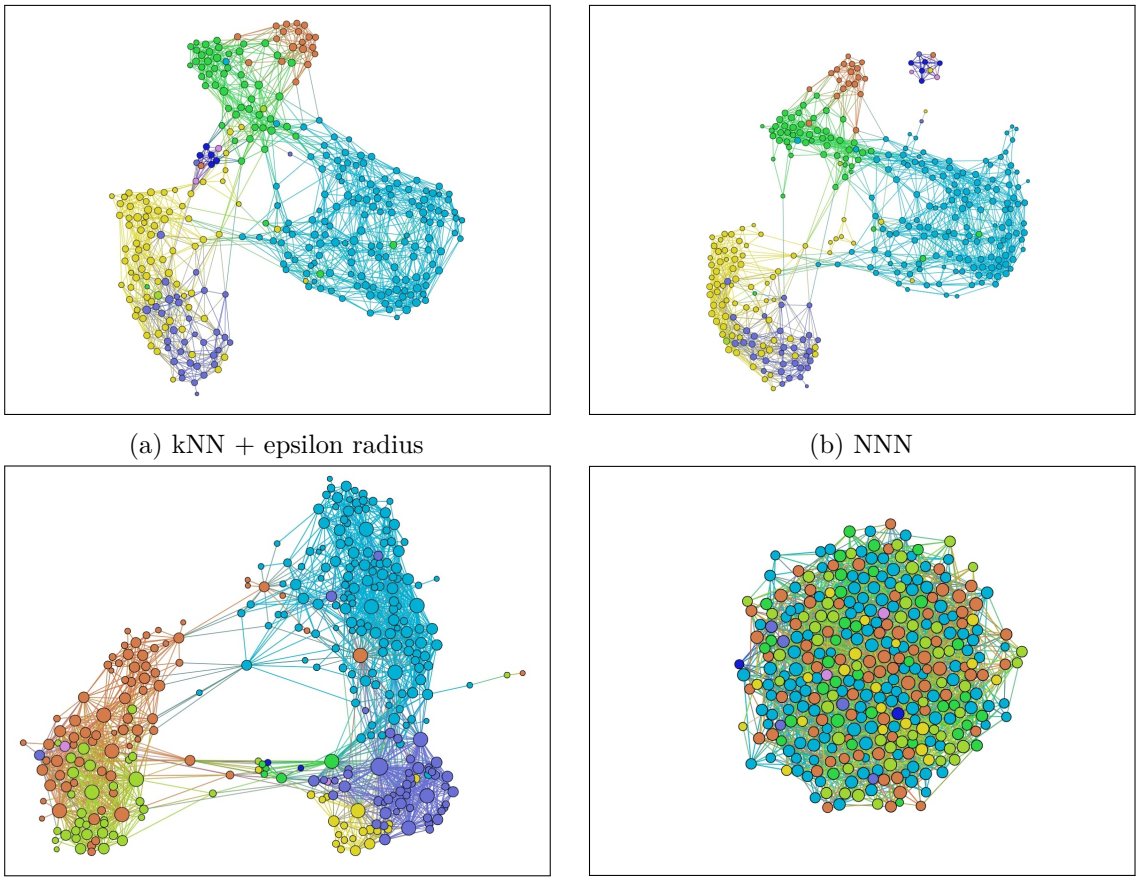
(c) LRNet

(d) Erods Renyi

Obr. 11: Siete datasetu Iris

### 7.1.2 E.Coli dataset

Dataset obsahuje 336 meraní proteinových sekvencií baktérie Escherichia coli. U každého merania bolo sledovaných 8 atribútov, na základe ktorých bola meraniu priradená 1 z 8 tried reprezentujúca, v ktorej časti bunky sa proteín nachádza. [19]



(a) kNN + epsilon radius

(b) NNN

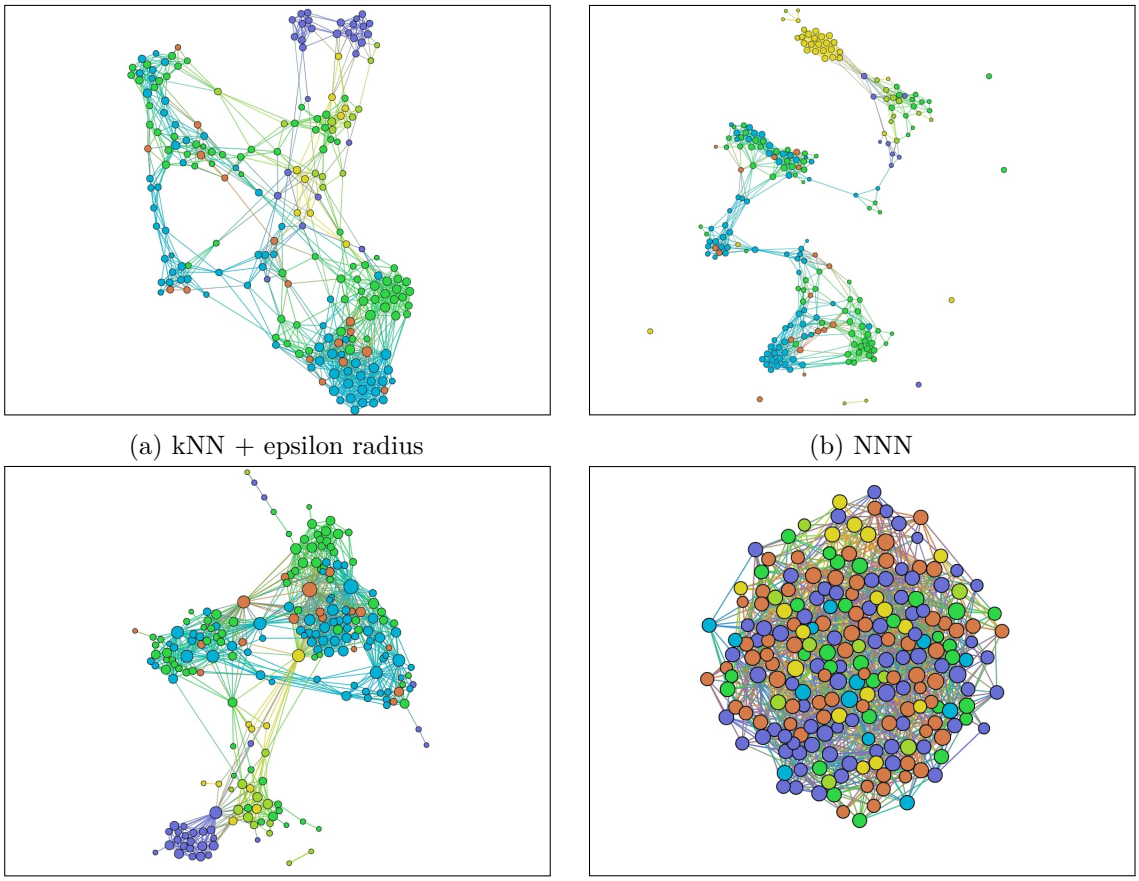
(c) LRNet

(d) Erods Renyi

Obr. 12: Siete datasetu E.Coli

### 7.1.3 Glass dataset

Dataset obsahuje 214 meraní vlastností skla. Pri každom zázname bolo meraných 9 vlastnosti, na základe ktorých bol určený 1 zo 7 tipov skla. [20]



(a) kNN + epsilon radius

(b) NNN

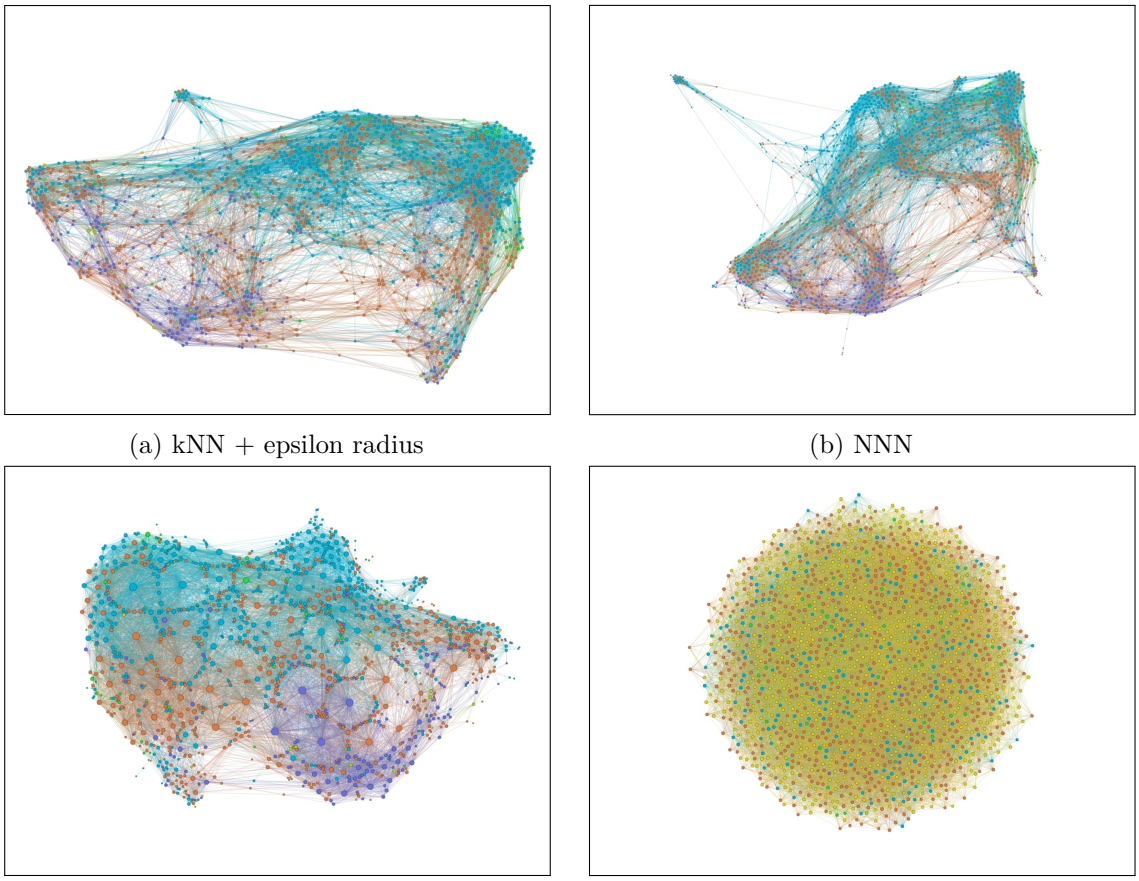
(c) LRNet

(d) Erods Renyi

Obr. 13: Siete datasetu Glass

### 7.1.4 Wine red dataset

Dataset obsahuje merania kvality červeného vína. Na základe 11 vlastnosti vína je určená jeho kvalita od 0 do 10. Dataset je tvorený 1599 záznamami. Počtom vlastností, záznamov a splývajúcimi triedami tento dataset predstavuje zaujímavý cieľ v oblasti klasifikácie. [21]



(a) kNN + epsilon radius

(b) NNN

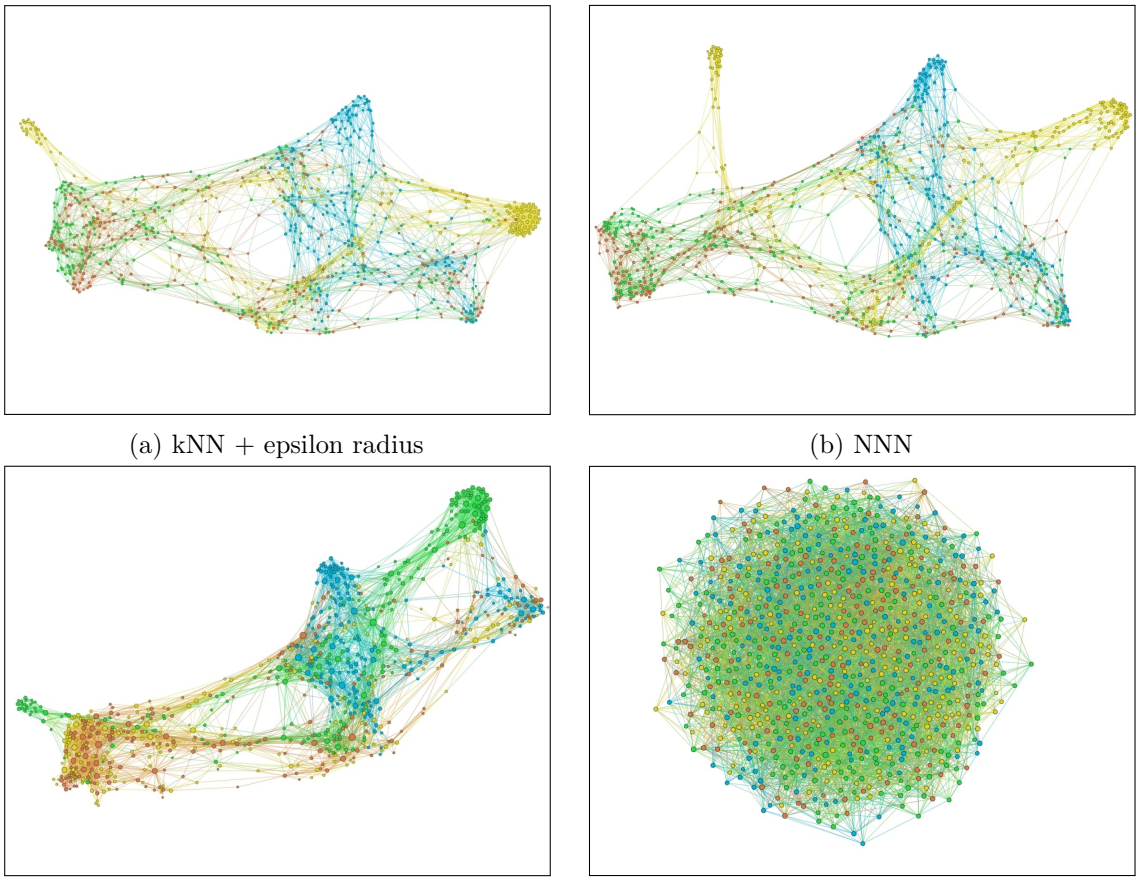
(c) LRNet

(d) Erods Renyi

Obr. 14: Siete datasetu Wine red

### 7.1.5 Vehicle dataset

Dataset obsahuje 4 triedy automobilov, ktorých tvar je zakódovaný do 17 atribútov.[22]



(a) kNN + epsilon radius

(b) NNN

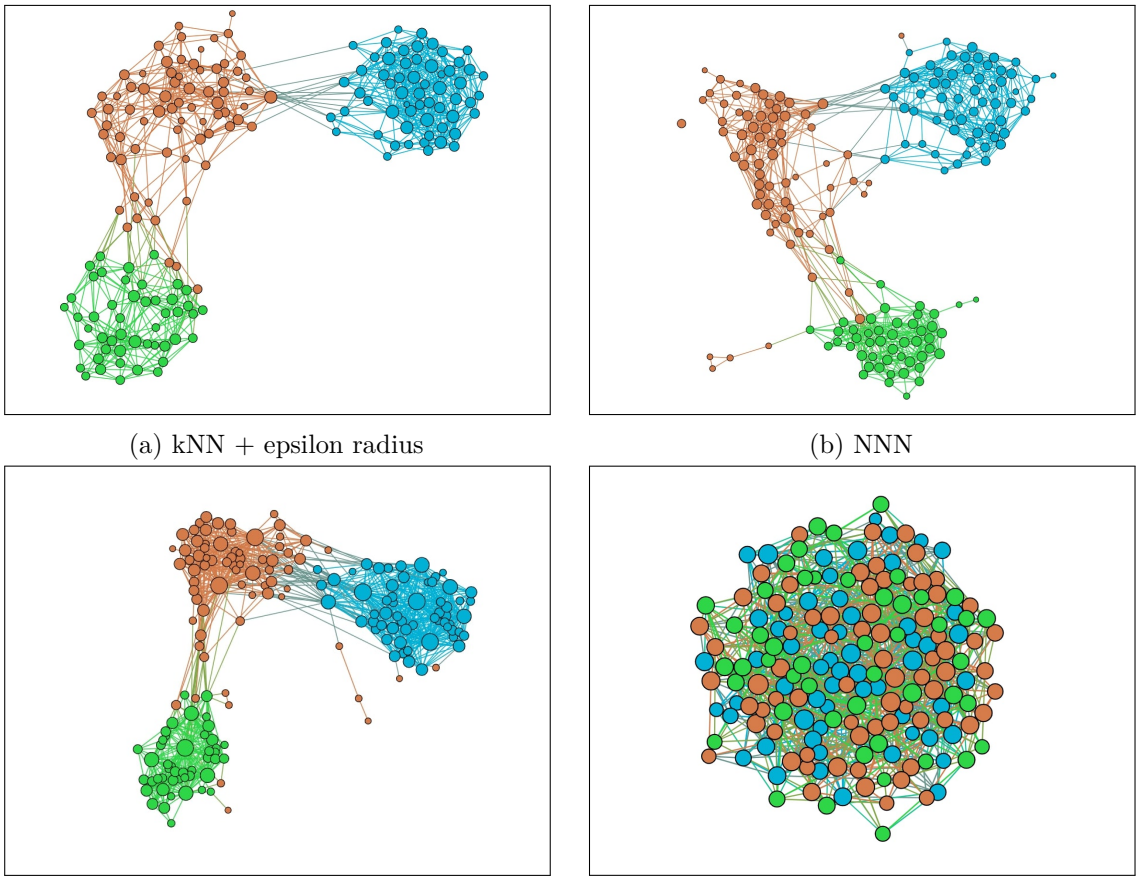
(c) LRNet

(d) Erods Renyi

Obr. 15: Siete datasetu Vehicle

### 7.1.6 Wine dataset

Dataset Wine obsahuje merania vlastností vína 3 odrôd pestovaných v rovnakej oblasti. 178 meraní je rozdelených do 3 rovnomerne veľkých a dobre oddelených triedy, čím sa do veľkej miery podobá datasetu Iris. Hlavný rozdiel je však v počte atribútov, ktorých Wine dataset obsahuje 13, vďaka čomu môže priniesť zaujímavé výsledky v porovnaní s datasetom Iris. [21]



(a) kNN + epsilon radius

(b) NNN

(c) LRNet

(d) Erods Renyi

Obr. 16: Siete datasetu Wine

## 7.2 Kvalita sieti generovaných jednotlivými algoritmami

Prvý experiment sa zameriava na porovnanie kvality sieti generovaných vyššie uvedenými algoritmami. Na experimenty bolo zvolených niekoľko typických datasetov na klasifikáciu uvedených v predchádzajúcej podkapitole 7.1. Pre každý dataset boli vygenerované siete s rovnakou hustotou. Referenčná hustota bola spočítaná na sieti generovanej algoritmom LRNet, ktorý bol použitý so základným nastavením. Na nastavenie ostatných algoritmov tak, aby mali približné rovnakú hustotu, bola využitá aplikácia, ktorá je súčasťou tejto práce. Pre každú sieť bola spočítaná presnosť zaradenia vrcholu do sieťe. Na tento účel boli použité 2 príbuzné miery uvedené v článku zameranom na algoritmus LRNet. [6]

Prvá miera s názvom *vážená presnosť zradenia vrcholu (VPZV)* je spočítaná ako priemerná presnosť zaradenia vrcholov. Presnosť zaradenia vrcholu je určená ako pomer súčtu váh hrán so susediacimi vrcholmi rovnakej triedy $w_{pos}$ a súčtu váh hrán všetkých susedov $w_{all}$.

$$w = \frac{w_{pos}}{w_{all}} \tag{17}$$

Druhá miera je vypočítaná z predchádzajúcej a bola pomenovaná ako *nevážená presnosť zaradenia vrcholu (NPZV)*. V tejto miere je presnosť zaradenia vrcholu binárna. Presnosť nadobúda hodnotu $w = 1$ v prípade, že súčet váh hrán s vrcholmi rovnakej triedy je najväčší, inak nadobúda hodnotu $w = 0$.

### 7.2.1 Výsledky experimentu

Pre lepšie vyhodnotenie výsledkov oboch metrík je v tejto práci uvedená tabuľka z článku [15], kde boli testované rovnaké datasety ako v tejto práci.

| Dataset | Proposed ($\alpha$,$\eta$) | kAOG | kNN ($k$) | WkNN ($k$) | PkNN ($p$) | C4.5 ($cf$,$m$) | MSVM ($cp$,$\gamma$) | NFL |
|---|---|---|---|---|---|---|---|---|
| Zoo | 96.2 ± 0.1 (7.8,1) | **97.0 ± 5.2** | 96.1 ± 5.9 (1) | 96.2 ± 5.8 (1) | 93.6 ± 7.1 (2) | 95.8 ± 5.3 (0.1,0) | 96.3 ± 6.4 ($2^{1}$,$2^{1}$) | **97.0 ± 4.7** |
| Hayes–Hoth | 57.0 ± 2.3 (3.5,1) | 55.7 ± 12.6 | 54.9 ± 12.4 (1) | 56.8 ± 13.2 (3) | 44.6 ± 11.3 (8) | 46.7 ± 10.5 (1,0) | 45.4 ± 13.1 ($2^{12}$,$2^{13}$) | **60.0 ± 9.7** |
| Iris | **98.1 ± 1.1** (3.1,29) | 97.4 ± 3.2 | 97.9 ± 3.4 (19) | 97.9 ± 3.3 (19) | 97.3 ± 3.2 (3) | 95.0 ± 5.8 (0.25,2) | 97.0 ± 4.6 ($2^{-2}$,$2^{3}$) | 95.7 ± 4.8 |
| Teaching | 63.1 ± 2.0 (6.0,1) | 62.5 ± 11.6 | 59.6 ± 10.2 (1) | 63.0 ± 12.3 (9) | 58.3 ± 9.0 (29) | 58.2 ± 14.9 (1,0) | 52.5 ± 7.9 ($2^{6}$,$2^{3}$) | **65.0 ± 11.3** |
| Wine | 84.6 ± 1.6 (9.4,1) | 83.5 ± 8.5 | 84.1 ± 8.5 (1) | 84.1 ± 8.2 (1) | 81.4 ± 11.9 (28) | 91.7 ± 6.7 (0.5,1) | **94.4 ± 5.8** ($2^{11}$,$2^{2}$) | 91.3 ± 6.0 |
| Image | 75.5 ± 0.8 (1.8,1) | 75.3 ± 7.5 | 75.3 ± 8.2 (1) | 75.4 ± 8.2 (3) | 75.5 ± 10.2 (16) | 80.7 ± 7.6 (0.8,3) | **86.7 ± 7.4** ($2^{10}$,$2^{-3}$) | 85.0 ± 6.9 |
| Glass | **72.5 ± 1.1** (10,2) | 72.5 ± 8.1 | 71.9 ± 8.6 (1) | 71.8 ± 9.0 (1) | 67.3 ± 11.8 (6) | 66.9 ± 9.4 (0.1,3) | 69.5 ± 5.6 ($2^{10}$,$2^{4}$) | 64.6 ± 8.5 |
| E.Coli | **87.5 ± 0.6** (4.4,9) | 85.8 ± 6.4 | 86.5 ± 5.2 (9) | 87.4 ± 5.4 (9) | 80.4 ± 5.2 (2) | 83.6 ± 6.1 (0.25,3) | 86.7 ± 8.2 ($2^{12}$,$2^{-9}$) | 80.0 ± 5.4 |
| Libras | 84.9 ± 0.8 (9.0, 1) | 85.4 ± 5.3 | 84.8 ± 5.5 (1) | 84.8 ± 5.4 (1) | 55.7 ± 4.2 (13) | 71.6 ± 7.5 (0.8,1) | 86.6 ± 5.0 ($2^{7}$,$2^{2}$) | **88.3 ± 5.1** |
| Balance | 96.3 ± 0.6 (4.1, 9) | 94.9 ± 2.5 | 94.7 ± 2.6 (1) | 96.7 ± 2.1 (11) | 76.6 ± 5.7 (8) | 89.6 ± 3.7 (0.5,1) | **98.2 ± 0.9** ($2^{7}$,$2^{0}$) | 97.5 ± 1.6 |
| Vehicle | 67.6 ± 0.6 (10.0, 6) | 67.9 ± 4.4 | 67.3 ± 4.0 (3) | 67.6 ± 4.1 (5) | 60.0 ± 5.5 (10) | 70.7 ± 3.5 (0.5,2) | **84.4 ± 3.4** ($2^{1}0$,$2^{3}$) | 75.3 ± 3.5 |
| Vowel | 97.7 ± 0.3 (2.8, 1) | **98.9 ± 0.7** | 97.8 ± 1.0 (1) | 98.8 ± 0.9 (11) | 96.6 ± 1.8 (8) | 78.6 ± 4.3 (0.5,0) | 97.5 ± 1.9 ($2^{7}$,$2^{0}$) | 98.8 ± 1.2 |
| Yeast | 59.5 ± 0.6 (9.6, 13) | 53.6 ± 3.8 | 58.7 ± 3.4 (15) | **60.9 ± 3.6** (16) | 48.0 ± 2.7 (1) | 55.8 ± 3.6 (0.1,5) | 58.9 ± 4.8 ($2^{11}$,$2^{0}$) | 52.4 ± 3.5 |
| Wine Q. (Red) | 62.3 ± 0.5 (6.4, 1) | 61.8 ± 3.6 | 61.3 ± 3.4 (1) | 64.0 ± 3.8 (19) | 38.9 ± 3.1 (27) | 59.8 ± 2.4 (1,0) | 60.4 ± 3.2 ($2^{9}$,$2^{1}$) | **65.2 ± 3.3** |
| Segment | 93.7 ± 0.2 (9.2, 1) | 93.7 ± 1.5 | 93.6 ± 1.6 (1) | 93.6 ± 1.4 (5) | 60.9 ± 3.2 (25) | 95.4 ± 1.2 (1,0) | 96.6 ± 1.2 ($2^{1}$1,$2^{0}$) | **97.0 ± 1.2** |
| MNIST | 96.3 ± 0.1 (10.0, 4) | 95.1 ± 1.7 | 95.3 ± 2.3 (1) | 95.7 ± 1.1 (5) | 94.3 ± 2.2 (25) | 94.5 ± 1.2 (1,0) | 96.4 ± 1.3 ($2^{1}$1,$2^{0}$) | **97.2 ± 0.7** |

Obr. 17: Prehľad výsledkov experimentov použitých v článku [15]

| **Algoritmus** | **Parametre algoritmu** | **vážená presnosť** | **nevážená presnosť** | **hustota siete** |
|---|---|---|---|---|
| Knn + epslion | k=9, eps=0.999 | **95.3** | **96.0** | 0.063 |
| NNN | n=13 | 93.3 | 94.7 | 0.062 |
| Lrnet | minVertex =1, reduction =1 | 91.0 | 95.3 | 0.063 |
| ErdosRenyi | p=0.0315 | 49.8 | 54.7 | 0.063 |

Tabuľka 1: Vlastnosti sieti pre Iris dataset

| **Algoritmus** | **Parametre algoritmu** | **vážená presnosť** | **nevážená presnosť** | **hustota siete** |
|---|---|---|---|---|
| Knn + epslion | k=5, eps=0.9843 | 86.3 | **79.9** | 0.38 |
| NNN | n=19 | **87.2** | 79.8 | 0.37 |
| Lrnet | minVertex =1, reduction =1 | 85.4 | 77.7 | 0.38 |
| ErdosRenyi | p=0.018 | 42.3 | 40.3 | 0.38 |

Tabuľka 2: Vlastnosti sieti pre E.Coli dataset

| **Algoritmus** | **Parametre algoritmu** | **vážená presnosť** | **nevážená presnosť** | **hustota siete** |
|---|---|---|---|---|
| Knn + epslion | k=10, eps=0.9965 | 61.6 | 64.5 | 0.05 |
| NNN | n=17 | **63.1** | **68.2** | 0.048 |
| Lrnet | minVertex =1, reduction =1 | 60.9 | 63.6 | 0.05 |
| ErdosRenyi | p=0.0245 | 36.9 | 36.4 | 0.05 |

Tabuľka 3: Vlastnosti sieti pre Glass dataset

| **Algoritmus** | **Parametre algoritmu** | **vážená presnosť** | **nevážená presnosť** | **hustota siete** |
|---|---|---|---|---|
| Knn + epslion | k=50, eps=0.99 | **51.9** | 57.0 | 0.032 |
| NNN | n=85 | 51.8 | **58.2** | 0.032 |
| Lrnet | minVertex =1, reduction =1 | 51.2 | 57.0 | 0.032 |
| ErdosRenyi | p=0.0245 | 44.9 | 45.2 | 0.032 |

Tabuľka 4: Vlastnosti sieti pre Wine red dataset

| **Algoritmus** | **Parametre algoritmu** | **vážená presnosť** | **nevážená presnosť** | hustota siete |
|---|---|---|---|---|
| **Knn + epslion** | k=10, eps=0.999 | **94.1** | **97.8** | 0.061 |
| **NNN** | n=17 | 92.9 | 96.1 | 0.062 |
| **Lrnet** | minVertex =1, reduction =1 | 90.5 | 95.5 | 0.062 |
| **ErdosRenyi** | p=0.031 | 52.8 | 61.2 | 0.059 |

Tabuľka 5: Vlastnosti sieti pre Wine dataset

| **Algoritmus** | **Parametre algoritmu** | **vážená presnosť** | **nevážená presnosť** | hustota siete |
|---|---|---|---|---|
| **kNN + epslion** | k=24, eps=0.99 | 62.9 | **69.1** | 0.028 |
| **NNN** | n=33 | **63.5** | 70.2 | 0.028 |
| **Lrnet** | minVertex =1, reduction =1 | 58.4 | 64.1 | 0.028 |
| **ErdosRenyi** | p=0.014 | 42.7 | 30.3 | 0.027 |

Tabuľka 6: Vlastnosti sieti pre Vehicle dataset

### 7.2.2 Zhodnotenie experimentu

Z výsledkov experimentu môžeme usúdiť, že presnosť oboch metrík, váženej presnosti zaradenia vrcholu a neváženej presnosti zaradenia vrcholu, sa pohybuje v rozsahu presnosti klasifikačných algoritmov uvedených v tabuľke 17. Taktiež môžeme pozorovať, že siete generované rôznymi algoritmami majú podobnú presnosť zaradenia vrcholov.

Výsledná presnosť poskytuje intuíciu, že algoritmy určené na prevod vektorových dát na sieť zachovávajú požadované vlastnosti dát. Výnimku tvorí sieť generovaná algoritmom *Erdos Renyi*. Tento algoritmus však nie je určený na prevod vektorových dát na sieť, čím upevňuje intuíciu o potrebe špecializovaných algoritmov.

Autor metrík neposkytuje ďalšie informácie o fungovaní vyššie uvedených metrík, ale vzhľadom na ich nízku výpočtovú náročnosť, oproti klasickým klasifikačným algoritmom, predstavujú priestor na skúmanie. Na základe týchto výsledkov boli vykonané dva experimenty. Prvý experiment môžeme zaradiť k exploatačnej analýze siete. Tento experiment bližšie skúma vplyv hustoty na presnosť zaradenia. Ďalšie experimenty sa zameriavajú na vplyv na rôzne faktory ovplyvňujúce presnosť klasifikácie.

## 7.3 Vplyv hustoty siete na presnosť zaradenia vrcholu

Ako bolo naznačené v predchádzajúcom experimente, cieľom tohoto experimentu je vytvoriť náhľad na to, ako vplýva hustota siete na výsledky presnosti zaradenia vrcholov, ktoré sú vypočítane pomocou vyššie uvedených metrík.

### 7.3.1 Príprava experimentu

Keďže aplikácia ani knižnice vytvorené v tejto prací neodhalili potrebu podobného využitia, boli najskôr implementované potrebné súčasti nutné na tento experiment. Vzhľadom na to, že potreba tejto funkcionality bola odhalená až vo fáze vykonávania experimentov, boli vytvorené len triedy pre vyhodnocovanie kvality sietí na rôzne nastavenia parametrov. Výsledky experimentov boli uložené do súborov vo formáte *.json*. Na vizualizáciu bola vytvorená statická webová stránka, ktorá súbory načíta z lokálneho úložiska a zobrazuje vo forme grafov. Detailnejší popis tried vytvorených na tento experiment je popísaný v kapitole 6.4.

### 7.3.2 Nastavenie parametrov

Pre všetky datasety použité v tejto práci boli použité rovnaké nastavenia konkrétnych algoritmov. Algoritmy LRnet a kNN + $\epsilon - radius$ majú dva konfigurovateľné parametre, ktoré do veľkej miery ovplyvňujú výslednú sieť. Hľadanie ideálnej kombinácie týchto parametrov je cieľom viacúčelovej optimalizácie, ktorá presahuje zameranie tejto práce, a preto boli použite algoritmy založené na náhodnom prehľadávaní okolia.

Je nutné podotknúť, že pri algoritme kNN + $\epsilon - radius$ , náhodná voľba medzi zmenou parametru $k$ alebo parametru $\epsilon$ môže spôsobovať väčšie nepresnosti. Tento problém je umocnený faktom, že algoritmus bol adaptovaný na použite podobnosti, ktorá je odlišná od typickej implementácie využívajúcej vzdialenosť, ktorá je citlivejšia na nastavenie parametrov. Použité podobnosti ovplyvňuje predovšetkým časť, v ktorej je použité $\epsilon$ okolie, čiže časť, v ktorej je vrchol spojený so všetkými susedami v $\epsilon$ okolí. Použitie $\epsilon$ okolia môže spôsobiť, že malá zmena parametru $\epsilon$ môže viesť k veľkému rozdielu v množstve použitých vrcholov, čo pri použití s podobnosťou *gaussian kernel* môže viesť ku skresleným výsledkom. Nesprávne nastavenie však nespôsobí úplne zlyhanie algoritmu, ale v najhoršom prípade algoritmus zdegraduje na izolovaný kNN alebo $\epsilon - radius$.

Pri algoritme LRNet nastáva podobný problém. Algoritmus LRnet však funguje na princípe základného nastavenia algoritmu poskytujúceho dobré výsledky a zlepšenie, respektíve zhoršenie môže byť dosiahnuté pomocou lokálneho prehľadávania v okolí základného nastavenia, čo prináša značnú výhodu oproti algoritmu kNN + $\epsilon - radius$.

Nastavenie parametrov experimentu pre rôzne algoritmy:

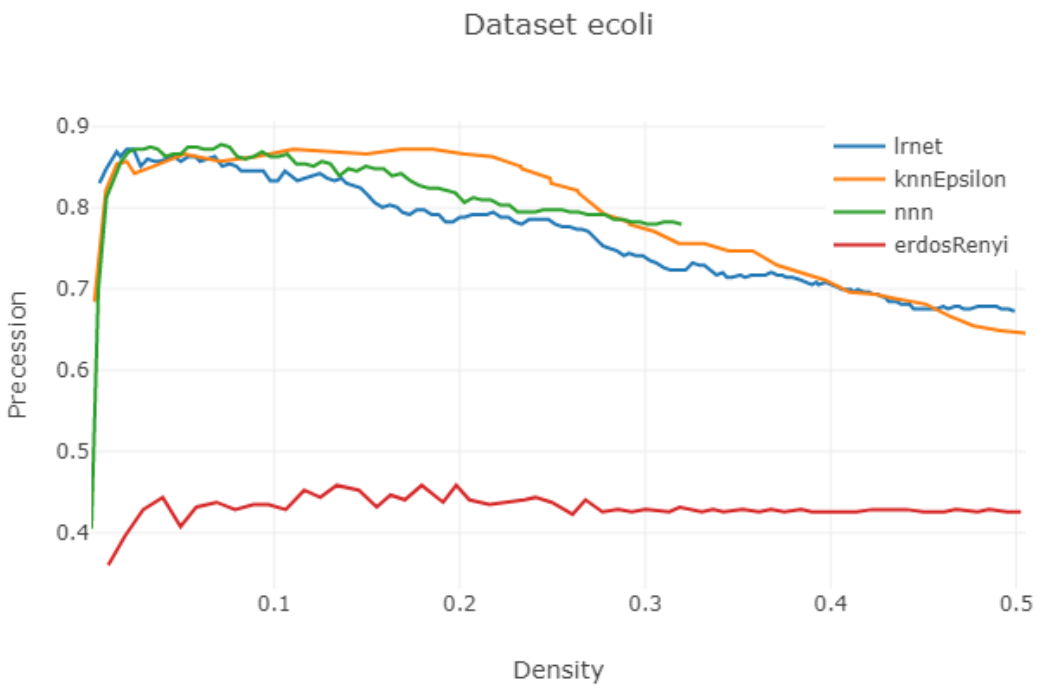
- kNN + $\epsilon - radius$:
  - **k**= 1
  - **krok k** = +2
  - **max k** = veľkosť najväčšej triedy
  - **epsilon** = 1.0
  - **krok epsilon** = - 0.005
- LrNet
  - **min deegres** =1
  - **krok min deegres** = +2
  - **max min deegres** = veľkosť najväčšej triedy
  - **reduction ratio** = 0.1
  - **krok reduction ratio** = + 0.1
- NNN
  - **n** = 1
  - **krok n** = +2
  - **max n** = veľkosť najväčšej triedy
- Erdos Renyi
  - **p** = 0.005

- **krok p** = 0.005
- **max p** = 1

Okrem nastavenia parametrov špecifických pre algoritmy bol nastavený parameter predčasného ukončenia ak hustota siete dosiahne 50%.

### 7.3.3 Výsledky a zhodnotenie experimentu

Na ilustráciu výsledkov bol zvolený dataset *Vehcile* a dataset *E.Coli*. Výsledky pre ostatné datasety sú umiestnené v prílohe A.




Obr. 18: Vplyv hustoty siete na kvalitu zaradenia pre dataset Ecoli


Obr. 19: Vplyv hustoty siete na kvalitu zaradenia pre dataset Vehicle

Z výsledkov experimentov môžeme usúdiť, že kvalita zaradenia vrcholov klesá s hustotou. Výnimku tvoria dataset Iris a Wine, ktoré tvoria veľmi dobre oddelené zhluky.Vďaka dobrej oddelenosti sú pri týchto datasetov vrcholy dobre zaradené aj pri relatívne vysokej hustote. Pri niektorých datasetoch môžeme vidieť, že výsledné siete vytvorené algoritmom kNN + $\epsilon - radius$ sú značne odlišné. Táto odlišnosť je najviac viditeľná pre dataset E.Coli, a preto bol vykonaný experiment, ktorý sa zameriava na skúmanie tohto výsledku. Z výsledkov, vo forme grafov, môžeme vidieť, že vrcholy sú najlepšie zradené pri veľmi nízkych hustotách grafov. Tento výsledok spôsobuje fakt, že pri nízkej hustote je sieť zvyčajne tvorená jedným veľkým komponentom a množstvom malých, často jednoprvkových, komponentov. Izolované, respektíve veľmi mále komponenty, sú pri **klasifikácii** nežiadané, a preto je nutné overiť množstvo a veľkosť komponentov a brať do úvahy len siete, ktoré majú nízky počet malých komponentov.

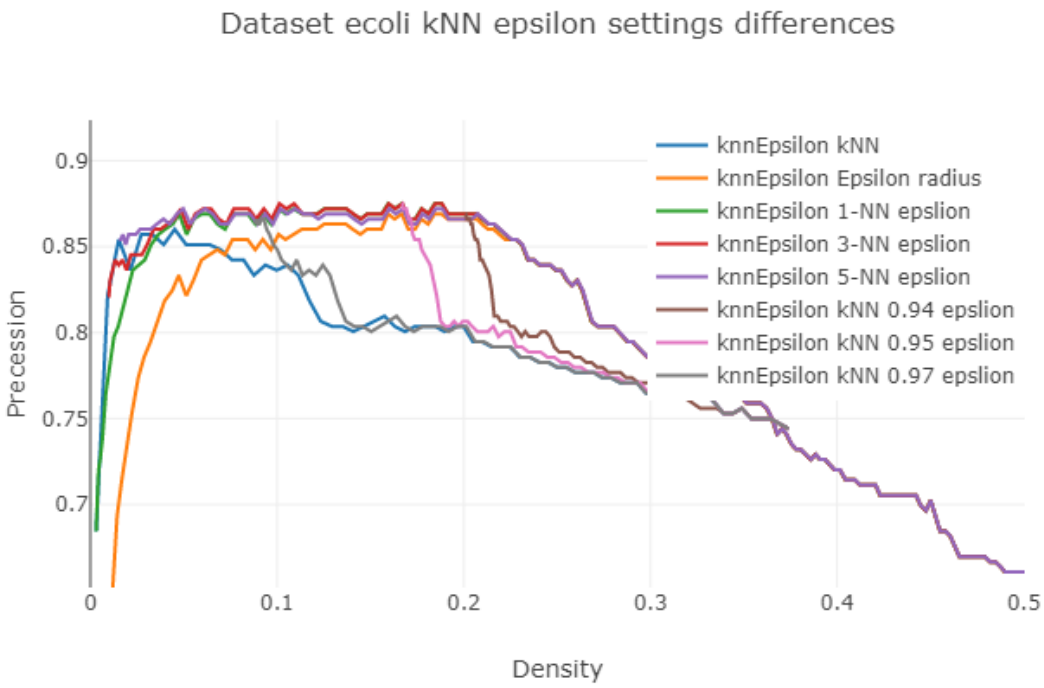
## 7.4 Detailný pohľad na algoritmus $kNN + \epsilon - radius$

Predchádzajúci experiment okrem iného odhalil odchýlky algoritmu kNN + $\epsilon - radius$ od výsledkov ostatných algoritmov. Toto správanie je najlepšie rozoznateľné z výsledkov prechádzajúceho experimentu pre dataset E.coli 20. Interval, kde tento algoritmus vytvára siete s lepším zaradením vrcholov, síce neobsahuje najlepší výsledok, ale aj tak predstavuje zaujímavú oblasť na preskúmanie. Cieľom tohto experimentu je zistiť, čo spôsobuje dané výsledky.

### 7.4.1 Nastavenie experimentu

Ako prvý krok experimentu bol algoritmus spustený s použitím jedeného parameteru pričom druhý parameter bol nastavený tak, aby nebol pri vytváraní siete použitý. V ďalšom kroku bolo zvolených niekoľko nastavení, pri ktorých izolované algoritmy poskytovali najlepšie výsledky. Následne bol algoritmus spustený so zafixovaným nastavením jedeného parametru.

### 7.4.2 Výsledky experimentu



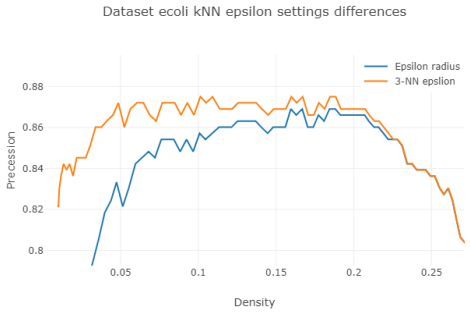

Obr. 20: Presnosť zaradenia vrcholov pre rôzne nastavenia algoritmu kNN + $\epsilon - radius$

Legenda na obrázku 20 naznačuje, ktorý parameter bol zafixovaný a s akým nastavením parametra. V prípade prvých dvoch meraní bol jeden z parametrov nastavený tak, aby sa pri vytváraní siete neprejavil a hybridný algoritmus bol redukovaný na jeden z dvoch jednoduchších algoritmov.

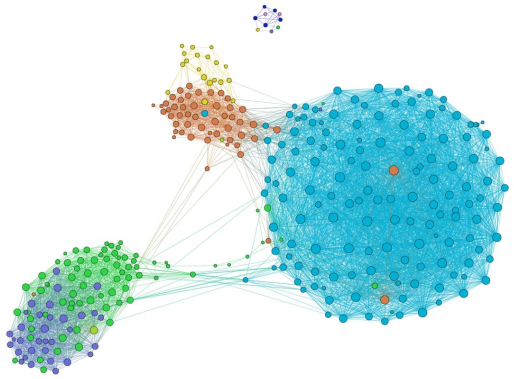
### 7.4.3 Zhodnotenie Experimentu

Výsledky experimentu ukázali, že odchýlka na obrázku 20 nebola spôsobená náhodou, ale vďaka dobrým výsledkom časti algoritmu, v ktorej je aplikovaný $\epsilon$ rádius. Ďalej môžeme pozorovať, že pri malých zmenách jedného alebo druhého parametra, algoritmus rýchlo konverguje ku kNN alebo k $\epsilon$ rádius. Pri detailnejšom pohľade na výsledky na obrázku 21 môžeme ďalej konštatovať,

že použitie hybridného algoritmu viedlo k zlepšeniu najlepšieho výsledku pre algoritmus epsilon o 1% a taktiež môžeme pozorovať, že použitie hybridného algoritmu viedlo k výraznému zlepšeniu pre riedke siete. Zo siete na obrázku 21 môžeme ďalej usúdiť, že v prípade malej zmeny parametrov budú hrany vytvárané prioritne medzi vrcholmi dobre oddelených komponentov, čo spôsobuje relatívne dlhý interval, na ktorom je tento výsledok stabilný.




(a) Porovnanie najlepšieho výsledku algoritmov epsilon rádius a kNN + $\epsilon - radius$



(b) Sieť (najredšia) pre najlepší výsledok zaradenia vrcholov

Obr. 21: Najlepšia presnosť zaradenia vrcholov E.Coli datasetu pre algoritmus kNN + $\epsilon - radius$

## 7.5 Porovnanie algoritmov pre klasifikáciu pomocou siete

V predchádzajúcich experimentov bola venovaná pozornosť predovšetkým skúmaniu sietí, ktoré boli vytvorené pomocou jednotlivých algoritmov. Výsledky ukázali, že siete s neúzkou hustotou zvyčajne poskytujú lepšie výsledky zaradenia vrcholov do siete.

Tento experiment sa zameriava na porovnanie výsledkov klasifikačných algoritmov *ease of access* a *average link weigth*. Najľahší spôsob určenia presnosti klasifikácie predstavuje takzvané mriežkové hľadanie. Tento spôsob je však výpočtovo veľmi náročný a v kombinácii s nastaveniami algoritmov na konštrukciu siete predstavuje veľmi neafektívne riešenie. Pri klasifikácii pomocou siete je najdôležitejším parameterom počet vrcholov, ktoré budú použité pri klasifikácii. Tento počet závisí od množstva parametrov ale predovšetkým od samotnej siete. Z tohoto dôvodu je vhodné použiť počet vrcholov na základe vlastnosti, ktorá sa dá spočítať priamo zo siete použitej na klasifikáciu. V tomto experimente je zvolenou vlastnosťou priemerný stupeň.

### 7.5.1 Príprava tréningovej siete

Pre porovnanie presnosti algoritmov bola zvolená 10-násobná krížová validácia. Dataset bol náhodne rozdelený do 10 skupín s rovnomerným zastúpením tried a postupne bola zvolená každá skupina ako testovacia kolekcia a z ostatných 9 skupín bola vytvorená tréningová sieť pomocou zvoleného algoritmu. Daný postup bol zopakovaný pre oba algoritmy 50 krát, aby sa tak vylúčila odchýlka merania.

### 7.5.2 Nastavenie parametrov

Nastavenie parametrov môžeme rozdeliť do 3 krokov.

1. **nastavenie podobnosti** - ovplyvňuje generovanú maticu podobnosti. Na tento experiment bola zvolená podobnosť *gaussian kernel* a dáta boli normalizované a reškalované do intervalu $< 0, 1 >$

2. **nastavenie konštrukcie siete** - parametre použitých algoritmov boli nastavené tak, aby generovali sieť s rovnakou hustotou. Na tento experiment boli využité nastavenia z kapitoly 7.2.

3. **nastavenie klasifikácie** - ako bolo naznačené, na určenie presnosti klasifikácie bola použitá k-násobná krížová validácia s parametrom $k = 10$ a samotný výsledok bol určený ako priemer 50 opakovaní klasifikácie s náhodne generovanými rozdeleniami tréningových a testovacích objektov.
   Algoritmus *Average Link Weigth* je bezparametrický, čiže ďalšie nastavenie nebolo potrebné. Algoritmus *Ease of Access* má 3 parametre. Pre pramater epsilon bola zvolená hodnota 10, pri ktorej boli experimentálne zistené [15] stabilne dobré výsledky pre rôzne datasety. Parameter na nastavenie počtu iterácii markovovho reťazca bol nastavený na hodnotu 10, pričom bolo použitá implementácia, ktorá ukončuje markovov reťazec predčasne, ak sa vektor s pravdepodobnosťami od poslednej iterácie nezmení. Posledný parameter určuje koľko susedov vrcholu má byť použitých na samotnú klasifikáciu. Tento parameter bol nastavený na priemerný stupeň siete, rovnako ako pre algoritmus *Average Link Weigth*.

### 7.5.3 Výsledky experimentu

| | **Iris** | **E.coli** | **Glass** | **Wine red** | **Vehicle** | **Wine** |
|---|---|---|---|---|---|---|
| **kNN + epislon** | 95.4 ±0.002 | 86.2 ±0.002 | 62.9 ±0.004 | 57.7 ±0.002 | 68.3 ±0.004 | 96.1 ±0.002 |
| **NNN** | 95.2 ±0.001 | 85.9 ±0.002 | 64.2 ±0.005 | 58.0 ±0.002 | 68.5 ±0.005 | **96.8 ±0.002** |
| **LRNet** | **96.1 ±0.002** | 86.5 ±0.002 | 60.2 ±0.006 | 57.4 ±0.002 | 68.4 ±0.006 | 95.7 ±0.002 |
| **ErdosRenyi** | 95.8 ±0.002 | **86.6 ±0.002** | **64.4 ±0.006** | **58.4 ±0.003** | **68.6 ±0.004** | 96.4 ±0.002 |

Tabuľka 7: Presnosť klasifikácie algoritmu Ease of access

| | Iris | E.coli | Glass | Wine red | Vehicle | Wine |
|---|---|---|---|---|---|---|
| **kNN + epislon** | **95.6 ±0.001** | 86.8 ±0.002 | 63.8 ±0.005 | 57.9 ±0.002 | 68.5 ±0.003 | **96.7 ±0.002** |
| **NNN** | 95.4 ±0.002 | **87.0 ±0.002** | 64.1 ±0.004 | **58.2 ±0.002** | 68.4 ±0.004 | 96.5 ±0.002 |
| **LRNet** | 95.2 ±0.003 | 86.9 ±0.002 | **64.2 ±0.005** | **58.2 ±0.002** | **68.6 ±0.004** | 96.4 ±0.003 |
| **ErdosRenyi** | 95.2 ±0.002 | 86.7 ±0.002 | 64.1 ±0.005 | 58.1 ±0.004 | 68.4 ±0.003 | 96.6 ±0.003 |

Tabuľka 8: Presnosť klasifikácie algoritmu Average link weight

Okrem výsledkov klasifikácie uvedených v tabuľkách 7 a 8 boli vypočítané aj konfidenčné intervaly. Výsledok ukázal, že pre väčšinu sietí sa 95% konfidenčný interval pohyboval medzi +- 0.001 a +- 0.003 bez ohľadu na použité algoritmy na zostrojenie siete a klasifikáciu. Výnimku tvorí iba dataset Glass, u ktorého sa 95% konfidenčný interval pohybuje medzi +- 0.004 a +- 0.007. Výsledky ukazujú veľmi dobrú stabilitu oboch klasifikačných algoritmov. Rozdiel medzi datasetom Glass a ostatnými datasetmi je zanedbateľný, a preto nebude ďalej skúmaný.

### 7.5.4 Zhodnotenie experimentu

Výsledky experimentu na prvý pohľad naznačujú, že siete s rovnakou hustotou, ktoré boli zostrojené rôznymi algoritmami, poskytujú rovnako dobré výsledky klasifikácie. Zaujímavý je aj výsledok klasifikácie pomocou siete zostrojenej technikou *Erdos Renyi*, kde je výsledok klasifikácie rovnaký ako pri použití špecializovaných algoritmov.

Po hlbšom skúmaní výsledku tohoto experimentu boli určené dve hlavné príčiny. Hlavnou príčinou bolo vytvorenie siete s rovnakou hustotou v kombinácii s použitím priemerného stupňa siete pri samotnej klasifikácii. Ak siete majú s rovnakou hustotu, potom majú aj rovnaký priemerný stupeň a výber vrcholov použitých pri klasifikácii je rovnaký.

Vzhľadom k vyššie uvedeným zisteniam bol experiment zopakovaný znova, pričom bola použitá odlišná stratégia voľby počtu vrcholov.

## 7.6 Vplyv počtu vrcholov na výsledok klasifikácie

Z výsledku prechádzajúceho experimentu vyplynulo, že ak sú pre klasifikáciu použité siete s rovnakou hustotou a ak je použitý priemerný stupeň pri samotnej klasifikácii, samotná sieť nemá na klasifikáciu vplyv.

Tento experiment sa zameriava na sledovanie vplyvu počtu vrcholov na výsledok klasifikácie. V predchádzajúcom experimente bolo okrem samotného výsledku zistené, že pri voľbe počtu vrcholov je priemerný stupeň nevhodný a špecifické vlastnosti sietí, ktoré sú generované rôznymi algoritmami, sa neprejavia. Ako ďalší krok skúmania vplyvu algoritmov na konštrukciu sietí

bol zopakovaný predchádzajúci experiment, s rozdielom pri voľbe počtu vrcholov použitých pri klasifikácii.

Na určenie počtu vrcholov, ktoré sú použité na klasifikáciu, boli opäť použité distribúcie vrcholov vo vygenerovaných sieťach. V tomto experimente bol však namiesto priemerného stupňa, stupeň z prvého kvartilu, mediánu a tretieho kvartilu. Použitie kvartilov bolo zvolené s intuíciou, že dáta reprezentujúce vlastnosti objektov reálneho sveta sú vystihnuté v sieti reálneho sveta (angl.*small-world networks*). Keďže siete reálneho sveta obsahuje veľa vrcholov s malým stupňom a malý počet vrcholov s veľmi vysokým stupňom, použitie kvartilov by malo mať vplyv predovšetkým na algoritmy generujúce tento typ sietí.

Na uskutočnenie tohto algoritmu bolo nutné upraviť klasifikátor *Average Link Weight*, ktorý bol implementovaný ako bezparametrický a pri výbere vrcholov používal priemerný stupeň. Modifikácia spočívala v pridaní parametru určujúceho stratégiu pre výber počtu vrcholov. Pre potrebu experimentu boli implementované 4 stratégie:

1. **Q1** - použitie prvého kvartilu
2. **Median** - použite mediánu
3. **Q3** - použite tretieho kvartilu,
4. **Mean** - pôvodná stratégia, ktorá využíva priemerný stupeň

### 7.6.1 Výsledky experimentu

Skratky: **EoA** - Ease of Access, **ALW** - Average Link Weight

Podobne ako pri predchádzajúcom experimente boli vypočítané 95% konfidenčné intervaly, ktorých výsledok bol rovnaký, čiže počet vrcholov použitých na klasifikáciu nemá vplyv na stabilitu algoritmov.

| **EoA** | **Q1** | **Median** | **Q3** |
|---|---|---|---|
| **kNN + epsilon** | 95.5 ±0.002 | 95.6 ±0.002 | 95.6 ±0.002 |
| **NNN** | 95.2 ±0.002 | 95.1 ±0.002 | 95.3 ±0.002 |
| **LRNet** | 95.5 ±0.003 | 96.2 ±0.002 | 95.8 ±0.002 |
| **ErdosRenyi** | **95.7 ±0.003** | 96.0 ±0.002 | 95.8 ±0.002 |

| **ALW** | **Q1** | **Median** | **Q3** |
|---|---|---|---|
| **kNN + epsilon** | 94.5 ±0.002 | 95.5 ±0.002 | 95.6 ±0.002 |
| **NNN** | 96.0 ±0.002 | 95.5 ±0.002 | 95.6 ±0.002 |
| **LRNet** | 95.2 ±0.002 | 96.4 ±0.002 | **96.2 ±0.002** |
| **ErdosRenyi** | **96.2 ±0.002** | 95.4 ±0.002 | 95.8 ±0.002 |

Tabuľka 9: Presnosť klasifikácie Iris datasetu

| EoA | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | **87.4 ±0.002** | 87.3 ±0.002 | 85.3 ±0.002 |
| **NNN** | 87.1 ±0.002 | 85.4 ±0.002 | 84.6 ±0.002 |
| **LRNet** | 86.4 ±0.003 | 87.0 ±0.002 | 84.5 ±0.002 |
| **ErdosRenyi** | 87.3 ±0.002 | 86.8 ±0.002 | 86.2.3 ±0.002 |

| ALW | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 86.8 ±0.002 | 87.7 ±0.002 | 86.1 ±0.002 |
| **NNN** | **87.9 ±0.002** | 86.2 ±0.002 | 85.5 ±0.002 |
| **LRNet** | 86.81 ±0.002 | 87.7 ±0.002 | 84.8 ±0.002 |
| **ErdosRenyi** | 87.3 ±0.002 | 86.8 ±0.002 | 86.2 ±0.002 |

Tabuľka 10: Presnosť klasifikácie E.Coli datasetu

| EoA | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 63.1 ±0.005 | 62.8 ±0.006 | 63.0 ±0.005 |
| **NNN** | **66.6 ±0.004** | 63.2 ±0.005 | 64.8 ±0.006 |
| **LRNet** | 63.6 ±0.006 | 60.4 ±0.005 | 59.3 ±0.006 |
| **ErdosRenyi** | 65.0 ±0.007 | 64.8 ±0.005 | 62.8 ±0.005 |

| ALW | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 64.3 ±0.004 | 63.8 ±0.006 | 64.1 ±0.004 |
| **NNN** | 66.7 ±0.003 | 63.9 ±0.005 | 64.1 ±0.005 |
| **LRNet** | **70.6 ±0.004** | 64.7 ±0.005 | 62.0 ±0.004 |
| **ErdosRenyi** | 64.6 ±0.005 | 64.5 ±0.005 | 62.0 ±0.004 |

Tabuľka 11: Presnosť klasifikácie Glass datasetu

| EoA | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 96.1 ±0.002 | 96.1 ±0.002 | 96.3 ±0.002 |
| **NNN** | 96.4 ±0.002 | 96.6 ±0.002 | 97.2 ±0.002 |
| **LRNet** | 95.7 ±0.003 | 95.7 ±0.003 | 97.0 ±0.001 |
| **ErdosRenyi** | 95.6 ±0.002 | 96.5 ±0.003 | **97.5 ±0.002** |

| ALW | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 96.5 ±0.002 | 96.4 ±0.003 | 96.4 ±0.002 |
| **NNN** | 96.2 ±0.003 | 96.9 ±0.002 | 97.3 ±0.002 |
| **LRNet** | 96.0 ±0.002 | 96.0 ±0.003 | **97.5 ±0.001** |
| **ErdosRenyi** | 96.0 ±0.002 | 96.6 ±0.003 | 97.3 ±0.002 |

Tabuľka 12: Presnosť klasifikácie Wine datasetu

| EoA | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 68.0 ±0.004 | 68.6 ±0.004 | 68.2 ±0.003 |
| **NNN** | **69.3 ±0.005** | 68.4 ±0.004 | 67.5 ±0.011 |
| **LRNet** | 69.1 ±0.002 | 68.2 ±0.003 | 67.7 ±0.004 |
| **ErdosRenyi** | 69.1 ±0.008 | 68.9 ±0.002 | 67.6 ±0.008 |

| ALW | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 68.3 ±0.002 | 68.3 ±0.004 | 68.3 ±0.007 |
| **NNN** | 68.9 ±0.006 | 67.8 ±0.003 | 67.2 ±0.004 |
| **LRNet** | **70.2 ±0.008** | 69.1 ±0.008 | 67.0 ±0.005 |
| **ErdosRenyi** | 68.3 ±0.008 | 68.2 ±0.006 | 67.9 ±0.007 |

Tabuľka 13: Presnosť klasifikácie Vehicle datasetu

| EoA | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 57.7 ±0.002 | 57.8 ±0.002 | 57.8 ±0.001 |
| **NNN** | 59.0 ±0.002 | 57.8 ±0.002 | **59.1 ±0.002** |
| **LRNet** | 57.6 ±0.003 | 57.6 ±0.002 | 57.5 ±0.002 |
| **ErdosRenyi** | 58.4 ±0.003 | 58.4 ±0.003 | 58.8 ±0.002 |

| ALW | Q1 | Median | Q3 |
|---|---|---|---|
| **kNN + epsilon** | 57.8 ±0.003 | 57.6 ±0.003 | 58.0 ±0.002 |
| **NNN** | 58.9 ±0.003 | 58.1 ±0.003 | 58.7 ±0.002 |
| **LRNet** | **59.4 ±0.002** | 58.7 ±0.004 | 58.1 ±0.002 |
| **ErdosRenyi** | 58.7 ±0.003 | 58.7 ±0.003 | 58.3 ±0.002 |

Tabuľka 14: Presnosť klasifikácie Wine red datasetu

### 7.6.2 Zhodnotenie experimentu

Výsledky experimentu naznačujú, že pri použití kvartilov sa jednotlive výsledky začínajú líšiť na základe použitého algoritmu. Pri dôkladnejšom zameraní sa na jednotlive datasety môžeme taktiež pozorovať, že pri niektorých datasetoch presnosť s množstvom použitých hrán (na základe použitej stratégie) klesá, pričom pri iných datasetoch stúpa. Výnimku tvorí len Iris dataset, u ktorého je nutné dodať, že normalizácia, ktorá bola použitá pre všetky datasety, nie je vhodná a mohla skresliť výsledky.

# 8 Diskusia

V práci bolo vykonaných niekoľko experimentov, ktorých cieľom bolo preskúmať siete vytvorené rôznymi algoritmami a následné použitie vytvorených sietí na klasifikáciu.

Prvá časť experimentov, zameraná na kvalitu sietí ukázala, že kvalitu sietí môžeme merať na základe presnosti zaradenia vrcholu do siete. Pre tento účel bola použitá miera definovaná v článku [6]. Experimenty ukázali, že presnosť zaradenia vrcholov vypočítaná pomocou tejto miery sa pohybuje v rozsahu výsledkov klasických klasifikačných algoritmov, vďaka čomu môžeme odhadnúť výsledok pred spustením klasifikácie. Oddelenie konštrukcie siete od klasifikácie umožnilo vytvorenie a ladenie siete nezávisle od použitého algoritmu na klasifikáciu. Presnosť zaradenia vrcholov bola testovaná na sieťach vytvorených pomocou algoritmov kNN + $\epsilon$ - rádius, NNN, LRNet a pomocou referenčného algoritmu na generovanie náhodných sietí Erdos Renyi. Výsledky ukázali, že algoritmy tvoria siete s podobne dobrým zaradením vrcholov, pričom najlepšie výsledky poskytujú riedke siete. Výnimku tvorí iba algoritmus Erdos Renyi, ktorý nie je určený pre prevod vektorových dát na sieť a zle výsledky sú očakávane. Napriek podobným výsledkom presnosti klasifikácie z experimentov vyplynulo niekoľko informácií o ostatných 3 algoritmoch.

1. algoritmus **NNN** má len 1 nastaviteľný parameter, ktorý je z množiny prirodzených čísel. Vďaka jednoduchému nastaveniu parametrov je algoritmus nenáročný na použite a hľadanie ideálnej siete na klasifikáciu je preto priamočiara.

2. algoritmus **kNN + $\epsilon$-radius** dokáže poskytnúť stabilne dobré výsledky na väčšom intervale pri niektorých typoch sietí. Algoritmus má však 2 nastaviteľné parametre, z ktorých jeden pochádza z množiny reálnych čísel a je extrémne citlivý na nastavenie. Ďalším problémom odhaleným v experimentoch je fakt, že nevhodné nastavenie pomeru medzi parametrami spôsobuje degradáciu hybridného algoritmu na jeden z 2 použitých algoritmov. Použitie a ladenie tohto algoritmu sa v experimentoch ukázalo ako obtiažne.

3. algoritmus **LRnet** zväčša mierne zaostáva za ostatnými dvoma algoritmami, ale tvorí siete, ktoré sa podobajú sieťam reálneho sveta, a preto je vhodnejší na vizuálnu analýzu, kde napriek zlému zaradeniu vrcholu do siete môže poskytnúť cennú informáciu, ak bol vrchol spojený s vrcholom s vysokým stupňom.

## 8.1 Ďalšie experimenty v oblasti prevodu vektorových dát na sieť

Algoritmy na prevod vektorových dát na sieť použité v práci boli transformované na použitie funkcie podobnosti, aj keď v základnej (publikovanej) verzii pracujú s funkciou odlišnosti. Dôvodom transformácie bolo použitie klasifikačných algoritmov v kombinácii s funkciou podobnosti *guassian kernel*. Zaujímavý experiment by preto mohol spočívať v použití inej funkcie

podobnosti alebo adaptácii algoritmov na prevod a klasifikáciu na použite odlišnosti, respektíve vzdialenosti.

Druhá časť experimentov ukázala, že počet vrcholov použitých pri klasifikácii ovplyvňuje výsledok pri oboch algoritmoch. Detailnejšie zhodnotenie výsledkov algoritmov ukazuje, že výsledky klasifikácie oboma algoritmami sú veľmi podobné. Pri porovnaní výsledkov jednotlivých algoritmov zohráva hlavnú úlohu rýchlosť. Z experimentov vykonaných v tejto práci vyplynulo, že algoritmus *Average Link Weight* je značne rýchlejší. Detailné skúmanie príčiny neprebehlo, ale môžeme predpokladať, že dôvodom pomalšieho behu algoritmu *Ease of Access* sú maticové operácie, ktorých sa v tomto algoritme používa viac. Experimenty ďalej ukázali, že siete generované pomocou algoritmu *Erdos Renyi* poskytujú výsledky klasifikácie porovnateľné s ostatnými algoritmami. Z tohto výsledku môžeme usúdiť, že zvolené metódy poskytujú dobré výsledky bez ohľadu na použitý algoritmus. Je však nutné dodať, že experimenty boli vykonané s cielene rovnakou hustotou, s použitím normalizácie a rovnakého počtu hrán pre všetky algoritmy. Prehnaná generalizácia mohla preto spôsobiť, že špecifické vlastnosti, ktoré dokážu výsledok klasifikácie ďalej zlepšiť, neboli odhalené. Experimenty zamerané na tieto vlastnosti v práci neboli vykonané a predstavujú zaujímavú oblasť pre ďalšie skúmanie tejto problematiky.

## 8.2 Ďalšie experimenty v oblasti klasifikácie vektorových dát pomocou sieti

Experimenty ukázali že počet vrcholov použitých na klasifikáciu dokáže ovplyvniť presnosť výsledkov. Pre počet vrcholov boli použité základne štatistické metódy, ktoré nepreukázali jednoznačný trend. Ďalšie experimenty by preto mohli túto oblasť preskúmať ešte hlbšie. Vzhľadom na to, že všetky algoritmy na konštrukciu siete (s výnimkou Erdos Renyi) používajú v nejakej podobe $n$ najbližších susedov, by mohlo byť toto číslo použité aj v kroku klasifikácie.

Algoritmus *Ease of Access* používa v kroku propagácie triedy z okolitých vrcholov dopredu určený počet vrcholov. Hrany s vrcholmi sú zoradené a následne je zvolená trieda klasifikovaného vrcholu na základe najväčšieho zastúpenia tried. Modifikácia algoritmu by mohla spočívať v adaptácia metódy použitej v *Average Link Weight*, kde je výsledná trieda určená na základe súčtu hrán.

Bez ohľadu na možnú prehnanú generalizáciu v experimentoch vykonaných v tejto práci môžeme usúdiť, že napriek faktu, že klasifikácia pomocou sietí neprekonáva výsledky klasických algoritmoch (aj keď sa im vo väčšine prípadoch blíži), poskytuje reprezentáciu dát vo forme siete čo v prípade vhodne zvoleného algoritmu na prevod vektorových dát na sieť poskytuje náhľad na štruktúru dát, ich vzájomnú interakciu a detailnejší pohľad na výsledok klasifikácie.

# 9 Záver

Cieľom práce bolo zoznámiť sa s problematikou v oblasti klasifikácie vektorových dát pomocou siete a následné aplikovanie získaných vedomosti k vytvoreniu aplikácie umožňujúcej klasifikácii vykonať a zhodnotiť jej výsledky.

Prvá časť práce sa venuje predovšetkým samotnému zaradeniu problematiky do oblasti strojového učenia a teoretickému zázemiu nutnému k transformácii nespracovaných dát do formy vhodnej na klasifikáciu pomocou siete. Medzi tieto kroky patria: načítanie dát z dátového zdroja a ich uloženie do vhodnej dátovej štruktúry v pamäti počítača, vyčistenie dát a prípadná normalizácia.

Ďalšia kapitola je venovaná teoretickému zázemiu problematiky prevodu vektorových dát na sieť spolu s uvedením algoritmov použitých v práci. V poslednej kapitole venovanej teoretickému zázemiu je predstavená problematika klasifikácie pomocou sieti spolu s popisom klasifikačných algoritmov použitých v práci.

Po teoretických kapitolách nasleduje kapitola venovaná knižniciam, ktoré boli implementované v práci a ich použitiu v aplikácii, ktorá bola taktiež implementovaná ako súčasť práce. Aplikácia umožňuje efektívne načítanie *.csv* súborov do datasetu, základné úkony nutné pre danú problematiku, ako je normalizácia a škálovanie, spustenie klasifikácie a vizualizáciu výsledkov klasifikácie pomocou interaktívneho modelu siete.

Predposledná kapitola práce sa venuje experimentom spojeným s klasifikáciou pomocou sietí. Experimenty vykonané v práci sa venujú predovšetkým dvom oblastiam: 1. faktory ovplyvňujúcu kvalitu siete vytvorenej z vektorových dát 2. vplyv počtu vrcholov na výslednú presnosť klasifikácie Výsledky experimentov sú detailnejšie zhrnuté v kapitole 8.

V súhrne môžeme povedať, že táto práca poskytla náhľad na problematiku v oblasti klasifikácie vektorových dát pomocou sieti, nástroje nutné k skúmaniu danej problematiky a experimenty nad *"state of art"* algoritmami v danej oblasti. Hlavný prínos tejto práce spočíva v štúdiu zvolených algoritmov v danej oblasti a nástrojoch na vizualizáciu výsledku klasifikácie, vďaka ktorému môžu byť okrem výsledkov klasifikácie skúmané aj dôvody výsledkov s väčšou presnosťou ako pri klasických metódach klasifikácie.

# Literatura


1. ZHANG, Zixuan. *Understand Data Normalization in Machine Learning*. Towards Data Science, 2019-08. Dostupné tiež z: https://towardsdatascience.com/understand-data-normalization-in-machine-learning-8ff3062101f0.
2. SILVA, Thiago; ZHAO, Liang. *Machine Learning in Complex Networks*. 2016-01. ISBN 978-3-319-17290-3. Dostupné z DOI: 10.1007/978-3-319-17290-3.
3. SILVA, Thiago; ZHAO, Liang. Pixel Clustering by Using Complex Network Community Detection Technique. In: 2007-10, s. 925–932. Dostupné z DOI: 10.1109/ISDA.2007.4389726.
4. *Bayes error rate*. Wikimedia Foundation, 2019-08. Dostupné tiež z: https://en.wikipedia.org/wiki/Bayes_error_rate#cite_note-stat-1.
5. HUTTENHOWER, Curtis et al. Nearest Neighbor Networks: clustering expression data based on gene neighborhoods. *BMC Bioinformatics*. 2007, roč. 8, č. 1, s. 250. Dostupné z DOI: 10.1186/1471-2105-8-250.
6. *Computing and Combinatorics*. Dostupné tiež z: https://link.springer.com/book/10.1007/978-3-319-62389-4.
7. JEBARA, Tony; WANG, Jun; CHANG, S. Graph construction and b-matching for semi-supervised learning. *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*. 2009-01. Dostupné z DOI: 10.1145/1553374.1553432.
8. WANG, Fei; ZHANG, Changshui. Label propagation through linear neighborhoods. IEEE Trans Knowl Data Eng. *Knowledge and Data Engineering, IEEE Transactions on*. 2008-02, roč. 20, s. 55–67. Dostupné z DOI: 10.1109/TKDE.2007.190672.
9. CELIKYILMAZ, Asli; THINT, Marcus; HUANG, Zhiheng. A Graph-based Semi-Supervised Learning for Question-Answering. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, 2009-08, s. 719–727. Dostupné tiež z: https://www.aclweb.org/anthology/P09-1081.
10. LANCICHINETTI, Andrea; FORTUNATO, Santo. Consensus clustering in complex networks. *Scientific reports*. 2012-03, roč. 2, s. 336. Dostupné z DOI: 10.1038/srep00336.
11. SEGAL, E.; WANG, H.; KOLLER, D. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*. 2003-07, roč. 19, č. suppl$_1$, s. i264–i272. ISSN 1367-4803. Dostupné z DOI: 10.1093/bioinformatics/btg1037.
12. MACSKASSY, Sofus A.; PROVOST, Foster. A Simple Relational Classifier. In: *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003*. 2003, s. 64–76.

13. FALOUTSOS, Christos; MCCURLEY, Kevin; TOMKINS, Andrew. Fast discovery of connection subgraphs. In: 2004-01, s. 118–127. Dostupné z DOI: 10.1145/1014052.1014068.

14. MCDOWELL, Luke; GUPTA, Kalyan; AHA, David. Cautious Collective Classification. *Journal of Machine Learning Research*. 2009-12, roč. 10, s. 2777–2836. Dostupné z DOI: 10.1145/1577069.1755879.

15. CUPERTINO, Thiago; ZHAO, Liang; CARNEIRO, Murillo. Network-based supervised data classification by using an heuristic of ease of access. *Neurocomputing*. 2015-02, roč. 149, s. 86–92. Dostupné z DOI: 10.1016/j.neucom.2014.03.071.

16. BILLWAGNER. *Delegates - C Programming Guide*. Dostupné tiež z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/.

17. ERDÖS, P.; RÉNYI, A. On Random Graphs I. *Publicationes Mathematicae Debrecen*. 1959, roč. 6, s. 290.

18. MARSHALL, Michael. Dostupné tiež z: https://archive.ics.uci.edu/ml/datasets/Iris.

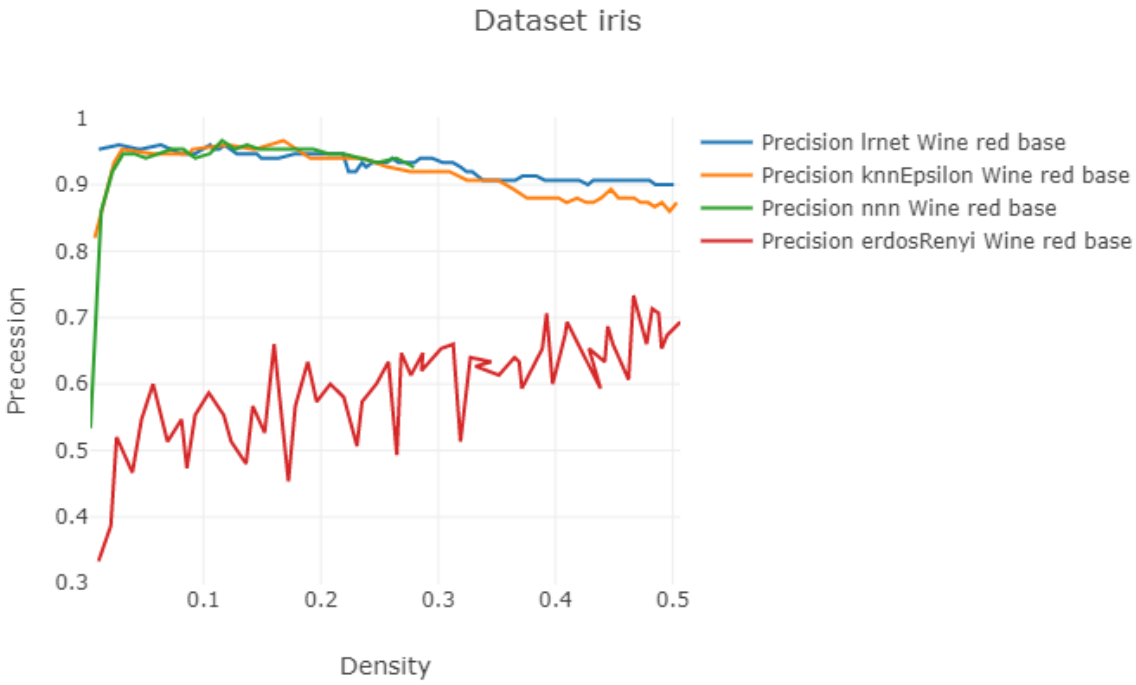
19. NAKAI, Kenta. Dostupné tiež z: https://archive.ics.uci.edu/ml/datasets/ecoli.

20. GERMAN, B. Dostupné tiež z: https://archive.ics.uci.edu/ml/datasets/Glass+Identification.

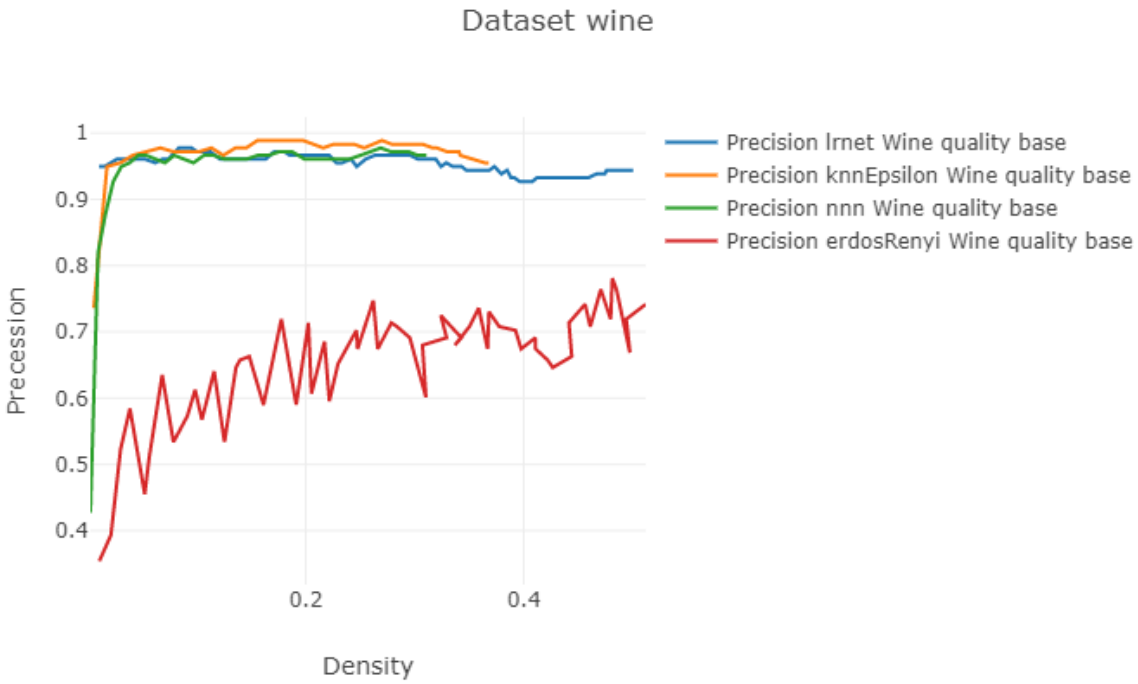
21. CORTEZ, Paulo. Dostupné tiež z: https://archive.ics.uci.edu/ml/datasets/wine+quality.

22. MOWFORTH, Drs.Pete; SHEPHERD, Barry. Dostupné tiež z: https://archive.ics.uci.edu/ml/datasets/Statlog+(Vehicle+Silhouettes).

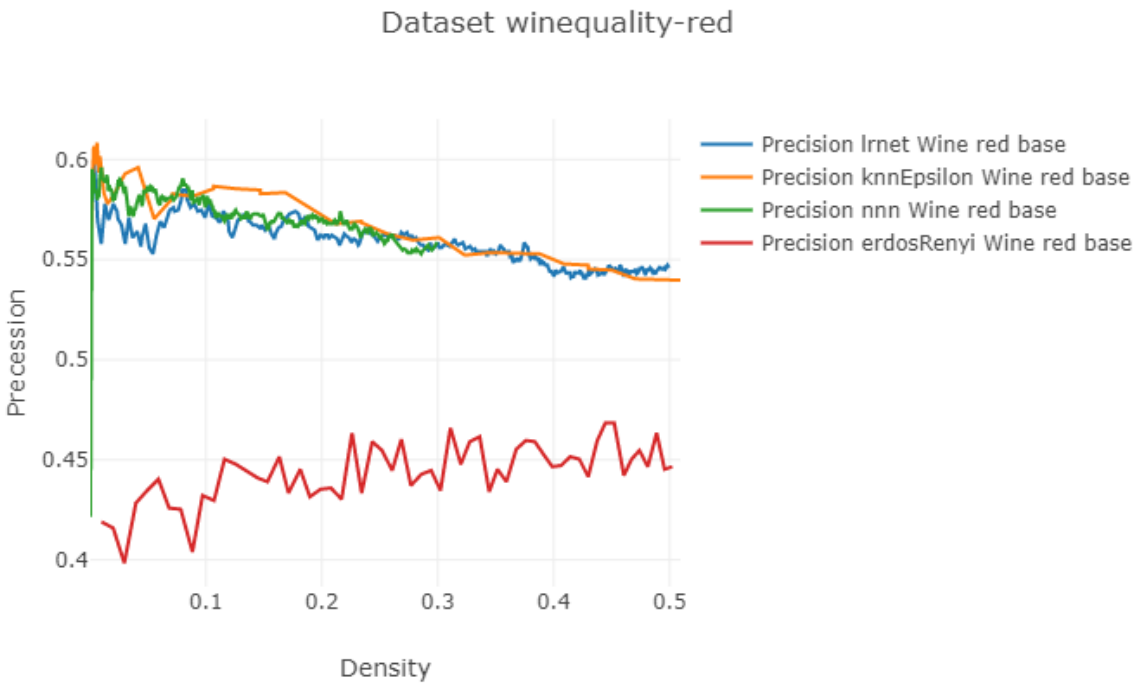
# A Vplyv hustoty na presnosť zaradenia vrcholov




Obr. 22: Vplyv hustoty siete na kvalitu zaradenia pre dataset Iris




Obr. 23: Vplyv hustoty siete na kvalitu zaradenia pre dataset Wine quality




Obr. 24: Vplyv hustoty siete na kvalitu zaradenia pre dataset Wine red


Obr. 25: Vplyv hustoty siete na kvalitu zaradenia pre dataset Glass